

Category Theory and Deep Learning

A Self-Contained Monograph

**Functors, Lenses, Optics, Backpropagation as a Natural Transformation
Polynomial Functors, Markov Categories, and the Yoneda Perspective
on Representation, Generalisation, and Robustness**

Silvano Salvador

2025–2026

Contents

Preface	vii
1 Categories, Functors, and Natural Transformations	1
1.1 The Idea of a Category	1
1.2 The Opposite Category and Duality	2
1.3 Functors: Maps Between Categories	3
1.4 Natural Transformations: Maps Between Functors	4
1.5 Isomorphisms, Equivalences, and Adjunctions	5
2 Monoidal Categories and String Diagrams	7
2.1 Composing Systems in Parallel	7
2.2 Cartesian and Compact Closed Categories	8
2.3 String Diagrams	9
2.4 Neural Networks as Morphisms in a Monoidal Category	10
3 Lenses, Para, and Backpropagation as a Functor	11
3.1 The Lens Pattern	11
3.2 Composition of Lenses	12
3.3 The Para Construction	13
3.4 Backpropagation as a Functor: The Cruttwell–Gavranović–Ghani–Wilson Theorem	13
4 Optics, Tambara Modules, and Automatic Differentiation	15
4.1 Beyond Lenses: The Need for Optics	15
4.2 Optics for Reverse-Mode Automatic Differentiation	16
4.3 Tambara Modules and the Abstract Theory of Optics	17
4.4 Mixed-Variance Functors and the Bidirectionality of Attention	18
5 Polynomial Functors and Neural Dynamical Systems	19
5.1 Polynomials as Wiring Diagrams	19
5.2 Dynamical Systems in Poly	20
5.3 The Mamba Architecture as a Polynomial Morphism	20
5.4 Polynomial Coalgebras and Memory	21
6 Categorical Probability and Markov Categories	22

6.1	The Problem of Probability in Category Theory	22
6.2	Conditional Independence and Bayesian Networks	23
6.3	Bayesian Inference as a Morphism	23
6.4	Variational Inference and the Evidence Lower Bound	24
7	Enriched Categories, Metric Spaces, and Robustness	26
7.1	Enrichment and the Quantification of Structure	26
7.2	Lawvere Metric Spaces	27
7.3	Enriched Functors as Robustness Certificates	27
7.4	Quantale-Enriched Categories and Generalised Metrics	28
8	The Yoneda Lemma and the Geometry of Representation	29
8.1	Representable Functors	29
8.2	The Yoneda Perspective on Embeddings	30
8.3	Presheaves and Universal Representations	30
8.4	The Neural Tangent Kernel as a Representable Functor	31
8.5	Attention as a Categorified Inner Product	31
9	Topoi, Sheaves, and the Logic of Generalisation	33
9.1	Grothendieck Topoi	33
9.2	The Internal Language of a Topos	34
9.3	Sheaves and Compositional Generalisation	34
9.4	Geometric Morphisms and Transfer Learning	35
10	Categorical Cybernetics and Open Games	36
10.1	Systems with Agency	36
10.2	Categorical Cybernetics	37
10.3	Gradient Descent as a Morphism in Categorical Cybernetics	37
11	Free Probability, the Composition of Random Layers, and the Freeness Conjecture	39
11.1	Classical vs. Free Independence	39
11.2	Categorical Consequences for Depth and Signal Propagation	40
12	Categorical Limits, Colimits, and Universal Approximation	41
12.1	Limits and Colimits in Category Theory	41
12.2	Kan Extensions and the Universal Property of Learning	42
13	Monads, Comonads, and the Algebra of Effects	43
13.1	Monads from Adjunctions	43
13.2	Kleisli Categories and Probabilistic Programs	44
13.3	The Writer and Reader Monads as Gradient Accumulators	44
13.4	Comonads, Dropout, and Structured Context	45
13.5	The Eilenberg-Moore Category and Weight Sharing	46
14	Fibred Categories, Dependent Types, and Variable-Type Learning	47

14.1	Fibrations and Families of Categories	47
14.2	Dependent Optics and Variable-Type Learning	48
14.3	The Sections Functor and Self-Supervised Learning	48
14.4	Slice Categories and Conditional Models	49
15	Double Categories, Traced Monoidal Structures, and Modular Architectures	50
15.1	Two Dimensions of Composition	50
15.2	Traced Monoidal Categories and Recurrence	51
15.3	Modular Architectures and Hypergraph Categories	52
15.4	The Calculus of Ends and Coends	52
16	Higher Categories, Coherence, and the Future of the Theory	54
16.1	The Need for Higher Structure	54
16.2	$(\infty, 1)$ -Categories and Homotopy-Coherent Backpropagation	54
16.3	Homotopy Type Theory and Formal Verification	55
16.4	Coherence as a Compiler Guarantee	56
16.5	Open Problems and Future Directions	57
17	Operads, Multicategories, and the Algebra of Network Wiring	58
17.1	Beyond Binary Composition	58
17.2	Algebras over Operads and Neural Architecture Families	59
17.3	The Operadic Composition of Attention Heads	59
17.4	Coloured Operads and Typed Wiring Diagrams	60
17.5	The Boardman-Vogt Tensor Product and Interchange	60
18	Equivariant Architectures and Categorical Symmetry	62
18.1	Symmetry in Machine Learning	62
18.2	Group Actions and the Category of G -Spaces	62
18.3	Linear Representations and Schur's Lemma Categorically	63
18.4	The Category of Equivariant Neural Networks	63
18.5	Group Convolution as a Natural Transformation	64
18.6	Steerable CNNs and Fibre Bundles	65
19	Information Geometry and the Natural Gradient Categorically	66
19.1	Statistical Manifolds and the Fisher Metric	66
19.2	Amari's Dual Connections	66
19.3	The Natural Gradient as a Riemannian Gradient	67
19.4	Categorical KL Divergence and the Information-Geometric Adjunction	68
19.5	The Categorical Cramér-Rao Bound	69
20	Categorical Signal Processing: Convolution, the Fourier Transform, and Spectral Methods	70
20.1	Convolution as a Morphism in a Monoidal Category	70
20.2	The Fourier Transform as a Monoidal Functor	71

20.3	Spectral Graph Theory and the Laplacian Functor	72
20.4	Wavelets and the Scattering Transform as a Functor	72
21	The Free Energy Principle, Active Inference, and Categorical Cybernetics of Self-Organisation	74
21.1	The Free Energy Principle	74
21.2	Categorical Formulation of Active Inference	74
21.3	The Markov Blanket and Category-Theoretic Self-Organisation	75
21.4	Connections to Predictive Coding and Deep Learning	76
21.5	Categorical Thermodynamics and the Free Energy Landscape	77
22	Loss Functions, Divergences, and the Geometry of Learning Objectives	78
22.1	Loss Functions as Morphisms	78
22.2	Bregman Divergences and Dually Flat Geometry	79
22.3	f -Divergences and the Data Processing Inequality	80
22.4	PAC-Bayes Bounds as Categorical Inequalities	80
23	Generative Models: GANs, Flows, and Diffusion as Categorical Structures	82
23.1	The Category of Generative Models	82
23.2	GANs as Morphisms in the Category of Open Games	83
23.3	Normalising Flows as Morphisms in Para(Diff)	85
23.4	Denoising Diffusion as a Markov Chain Morphism	86
23.5	Continuous-Time Diffusion and the SDE Category	88
23.6	The Unified View via Categorical Optimal Transport	88
24	Residual Architectures, Skip Connections, and Categorical Constructions	90
24.1	The Residual Connection as a Bimonoid Morphism	90
24.2	The Backward Pass through a Residual Connection	91
24.3	DenseNet, Pre-Activation, and Generalised Skip Connections	92
24.4	Neural ODEs as the Continuous Limit	92
24.5	Normalisation Layers and the Geometry of Activation Distributions	93
25	Categorical Causal Inference and the Do-Calculus	95
25.1	Structural Causal Models in the Markov Category	95
25.2	The Do-Calculus as a Category of Modified SCMs	96
25.3	Pearl's Do-Calculus as Rewriting Rules in BorelStoch	96
25.4	Counterfactuals, Twin Networks, and the Category of Possible Worlds	97
25.5	Causal Representation Learning and the Double Para Construction	98
26	Meta-Learning, MAML, and Higher-Order Categorical Structures	100
26.1	The Meta-Learning Problem as a 2-Categorical Optimisation	100
26.2	The Second-Order Backpropagation Functor	101
26.3	Few-Shot Learning and the Categorical Geometry of Class Representations	102

26.4	Hypernetworks as Morphisms in the Internal Category	103
27	The Categorical Theory of Optimisation	104
27.1	The Para Construction as an Optimisation Problem	104
27.2	The Geometry of Gradient Descent	104
27.3	Stochastic Gradient Descent and the Langevin Equation	105
27.4	Momentum Methods and Second-Order Geometry	106
27.5	Adam, Preconditioned Gradient Descent, and the Fisher Metric	107
27.6	The Loss Landscape: Saddle Points, Flat Minima, and the Morse-Theoretic Account	107
28	Graph Neural Networks and the Categorical Geometry of Relational Data	110
28.1	The Category of Graphs	110
28.2	Attributed Graphs and the Category $\mathbf{Grph}_{\mathcal{F}}$	111
28.3	Message Passing as a Natural Transformation	112
28.4	The Weisfeiler-Leman Test and the Limits of MPNN Expressivity	113
28.5	Spectral Graph Theory and the Graph Fourier Transform	113
28.6	Graph Pooling as a Pushforward in the Markov Category	115
28.7	Graph Transformers and Positional Encodings	115
29	Transformers and the Category of Sequence-to-Sequence Maps	117
29.1	The Transformer as a Composite Morphism	117
29.2	Self-Attention as Profunctor Composition	118
29.3	Positional Encodings and the Representation of Sequence Order	118
29.4	The Feed-Forward Network as Memory and Universal Approximator	119
29.5	Residual Connections and Layer Normalisation in the Transformer Block	120
29.6	Attention as a Kernel Method and Linearisation	120
29.7	Turing Completeness and the Categorical Universality of Transformers	121
30	Reinforcement Learning, Markov Decision Processes, and Categorical Dynamics	122
30.1	Markov Decision Processes as Polynomial Coalgebras	122
30.2	The Bellman Equation as a Fixed-Point in the Giry Monad	123
30.3	Deep Q-Learning and Parametric Bellman Residuals	123
30.4	Policy Gradient Methods and the Natural Policy Gradient	124
30.5	Actor-Critic as a Bidirectional Morphism	125
30.6	Reinforcement Learning from Human Feedback	125
31	Neural Scaling Laws and the Category of Model Families	127
31.1	Empirical Scaling Laws and Power-Law Behaviour	127
31.2	The Moduli Space of Parametric Model Families	127
31.3	Metric Entropy, Covering Numbers, and the Scaling Exponent	128
31.4	Phase Transitions and Emergent Capabilities	129
31.5	The Chinchilla Optimum as Categorical Optimal Transport	130

31.6 PAC-Bayes and the Information-Theoretic Lower Bound	130
32 Categorical Linear Algebra, Tensor Networks, and Quantum-Inspired Methods	132
32.1 Compact Closure, Traces, and the Partial Trace	132
32.2 Tensor Networks as String Diagrams in $\mathbf{FVect}_{\mathbb{R}}$	133
32.3 Tucker Decomposition and Convolutional Architecture	134
32.4 MERA and Hierarchical Representations	134
32.5 Quantum-Inspired Classical Algorithms	135
33 Categorical Semantics of Programming Languages and Differentiable Programming	137
33.1 The Curry-Howard-Lambek Correspondence	137
33.2 Linear Types and the SMCC	138
33.3 The Denotational Semantics of AD	138
33.4 Effects, Monads, and Side Effects in Differentiable Programs	139
33.5 Abstract Interpretation and the Categorical Semantics of Verification	140
Concluding Remarks	142
Exercises	145
Bibliographic Notes and References	149

Preface

Category theory entered mathematics in the 1940s through the work of Samuel Eilenberg and Saunders Mac Lane, who introduced it as a language for expressing the precise structural relationships between algebraic invariants in algebraic topology. For decades it remained largely a tool of pure mathematics — useful for homological algebra, algebraic geometry, and logic, but regarded by most as too abstract for applied science. Its arrival in the theory of computation came gradually: Lawvere’s categorical semantics in the 1960s and 1970s, the connections to lambda calculus and type theory through cartesian closed categories in the 1980s, and the discovery by Joyal and Street that string diagrams — informal graphical calculi used by physicists — could be given rigorous categorical foundations as the morphisms of (traced symmetric) monoidal categories.

What is new, and what motivates this monograph, is the discovery over the past decade that category theory provides not merely a language but a *theory* of deep learning: a framework in which the fundamental operations of machine learning — composition of layers, gradient-based parameter update, the forward and backward passes of backpropagation, probabilistic inference, and the expressivity of architectures — can be understood as instances of universal categorical constructions. This is not a metaphor or a rephrasing of existing ideas in new notation. It is the identification of genuinely new structure: the recognition, for instance, that a neural network with learnable parameters is a morphism in a *Para construction* over a suitable monoidal category; that the backward pass of backpropagation is a section of an *optic* fibred over the forward pass; that the chain rule of calculus is a consequence of the functoriality of the tangent functor; that Markov kernels between measurable spaces form a category with a statistical interpretation that subsumes both probabilistic graphical models and variational inference.

This monograph is self-contained. We assume comfort with linear algebra and real analysis, and some familiarity with neural networks at the level of understanding what a feedforward network is and what gradient descent does. We do not assume any prior knowledge of category theory; all categorical notions are developed from first principles, with attention to the intuition behind each definition and to its concrete manifestations in the deep learning context. The goal is not to catalogue every paper in the growing literature on categorical deep learning, but to develop the main conceptual threads with full rigour and to show, at each step, precisely why the categorical formulation is not merely equivalent to but *clearer and more powerful* than the standard one.

The structure of the exposition is as follows. We begin with the foundations — categories,

functors, and natural transformations — taking the time to develop many examples that will recur throughout, because the power of categorical reasoning lies entirely in the recognition of shared structure across different-seeming examples. We then build up the monoidal structure that is needed to talk about composition of systems, introducing string diagrams as a rigorous graphical calculus. The heart of the monograph is a detailed account of lenses and optics as the categorical structures underlying the bidirectionality of learning: the forward pass is the “get” component of a lens, and the backward pass is the “put”, and the coherence conditions on this pair are precisely what ensures that gradient updates are consistent. We then treat the Para construction, which parametrises morphisms by trainable weights, and show that the assembly of a deep network from layers is a functorial operation in the Para category. The chapter on automatic differentiation gives the sharpest categorical account of reverse-mode AD, connecting it to the theory of comonads and to the cotangent fibration. We turn then to polynomial functors, which provide a framework for understanding recurrent and stateful architectures as dynamical systems in a category of “wiring diagrams”. The final chapters treat categorical probability through the Markov category framework of Fritz, and close with the Yoneda lemma, which we interpret as a structural theorem about representation learning: every object is completely determined by its relationship to all other objects, a principle that underlies embedding methods, kernel machines, and the theory of attention.

Throughout, we prove what we claim. Where a full proof would require significant additional machinery, we give precise statements and sketch the key ideas, providing references for the complete arguments.

A note on notation. We write the composition of morphisms $f : A \rightarrow B$ and $g : B \rightarrow C$ as $g \circ f : A \rightarrow C$ (diagrammatic order is written $f;g$ when clarity demands it). We use calligraphic letters $\mathcal{C}, \mathcal{D}, \dots$ for categories, bold letters **Set**, **Vect**, $\dots$ for specific named categories, and Roman letters for objects and morphisms within a fixed category.

Chapter 1

Categories, Functors, and Natural Transformations

1.1 The Idea of a Category

A recurring pattern in mathematics is that the interesting objects of study — groups, vector spaces, topological spaces, smooth manifolds — are never studied in isolation. They are studied together with the *structure-preserving maps* between them: group homomorphisms, linear maps, continuous maps, smooth maps. The insight that is the starting point of category theory is that this pattern of “objects and maps” is itself a mathematical structure that can be isolated, axiomatised, and reasoned about. The axioms are few and simple, but their consequences, properly developed, reach very far.

Definition 1.1 (Category). A *category* $\mathcal{C}$ consists of:

1. a collection $\text{ob}(\mathcal{C})$ of *objects*,
2. for each pair of objects $A, B \in \text{ob}(\mathcal{C})$, a collection $\mathcal{C}(A, B)$ of *morphisms* (also called *arrows*) from A to B ,
3. for each triple of objects A, B, C , a *composition* operation

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \rightarrow \mathcal{C}(A, C), \quad (g, f) \mapsto g \circ f,$$

4. for each object A , a distinguished *identity morphism* $\text{id}_A \in \mathcal{C}(A, A)$,

satisfying:

1. (*Associativity*) for any morphisms $f : A \rightarrow B$, $g : B \rightarrow C$, $h : C \rightarrow D$,

$$h \circ (g \circ f) = (h \circ g) \circ f;$$

2. (*Unit laws*) for any $f : A \rightarrow B$, $\text{id}_B \circ f = f = f \circ \text{id}_A$.

The definition is spare, almost sparingly so. There is no mention of any structure on the morphisms beyond their ability to be composed. This apparent poverty is its strength: it

means that the theorems one proves from these axioms apply universally to every mathematical domain that fits the pattern.

Let us ground this at once in examples that will persist throughout this text.

The category **Set** has as its objects all sets (in some fixed Grothendieck universe, to avoid set-theoretic paradoxes) and as its morphisms the functions between sets. Composition is the usual composition of functions, and identities are identity functions. This is the most elementary category, and it will serve as a base of comparison for everything else.

The category **FVect**_{*k*} of finite-dimensional vector spaces over a field *k* (we will almost always take *k* = ℝ) has as objects the spaces ℝ^{*n*} for *n* ≥ 0, and as morphisms the linear maps. A linear map *f* : ℝ^{*m*} → ℝ^{*n*} is, of course, a matrix *W* ∈ ℝ^{*n*×*m*} once we fix bases, and composition of linear maps corresponds to matrix multiplication. The identity morphism on ℝ^{*n*} is the identity matrix *I*_{*n*}. This category is the native habitat of the linear layers of a neural network.

The category **Smooth** of smooth manifolds and smooth maps will appear when we discuss gradient flows and the differential structure of parameter spaces. The objects are smooth manifolds (including open subsets of ℝ^{*n*}, which is the case most relevant to the parameter spaces of neural networks), and the morphisms are smooth (*C*[∞]) functions.

A neural network itself can be thought of as a category in a degenerate sense. Consider a feedforward network with layers *L*₀, *L*₁, ..., *L*_{*n*}. We can form a category with objects {*L*₀, *L*₁, ..., *L*_{*n*}} and morphisms that are the (composed) network functions between layer activations: the only morphisms are the identity at each layer, the function *f*_{*i*} : *L*_{*i*−1} → *L*_{*i*} computed by the *i*-th layer, and their composites. This is a very thin category — it is essentially a directed path — but viewing it this way makes explicit that the compositional structure of a network is categorical in nature. Deeper networks correspond to longer composable chains of morphisms.

Two special cases of the category definition deserve explicit mention. A *monoid* is precisely a category with a single object. The single object is irrelevant; what matters is the collection of morphisms from that object to itself, together with composition (which is associative) and the identity. The monoid of natural numbers under addition is the one-object category with objects ℕ and morphisms {0, 1, 2, ...}, where *m* + *n* is the composite of *m* and *n*. A *group* is a monoid in which every morphism is invertible. A *partially ordered set* (poset) is a category in which there is *at most one* morphism between any two objects: if *a* ≤ *b* we write a morphism *a* → *b*, and transitivity of the order relation corresponds to composition. The identity at each object corresponds to reflexivity. These degenerate cases show that category theory genuinely subsumes large portions of algebra and order theory.

1.2 The Opposite Category and Duality

Every category *C* has an *opposite* (or *dual*) category *C*^{op}, obtained by formally reversing the direction of every morphism. Precisely: ob(*C*^{op}) = ob(*C*), *C*^{op}(*A*, *B*) = *C*(*B*, *A*), and

composition in $\mathcal{C}^{\text{op}}$ is defined by $g \circ_{\text{op}} f = f \circ_{\mathcal{C}} g$. Associativity and unit laws in $\mathcal{C}^{\text{op}}$ follow immediately from those in $\mathcal{C}$.

The importance of duality cannot be overstated: every theorem about categories has a *dual* theorem obtained by replacing $\mathcal{C}$ with $\mathcal{C}^{\text{op}}$, which amounts to reversing all arrows in every diagram. This means that every construction comes in pairs, and once you understand one side, the dual is “free”. We will see this repeatedly: limits and colimits, products and coproducts, the covariant and contravariant Yoneda lemma.

In the context of deep learning, the dual perspective already has immediate meaning. The forward pass of a network computes a morphism $f : X \rightarrow Y$ in some category of spaces. The backward pass, which propagates gradients from output to input, computes something that travels in the *opposite* direction: a morphism $X \leftarrow Y$ of some related quantity (the cotangent). This directional reversal is precisely the hallmark of a contravariant operation, and it is no coincidence that the categorical structures most naturally suited to describing backpropagation — lenses, optics — are built from pairs of morphisms, one going forward and one going backward.

1.3 Functors: Maps Between Categories

If categories are the objects of study, then functors are the maps between them. A functor is a structure-preserving map from one category to another.

Definition 1.2 (Functor). Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A *functor* $F : \mathcal{C} \rightarrow \mathcal{D}$ consists of:

1. an assignment $F : \text{ob}(\mathcal{C}) \rightarrow \text{ob}(\mathcal{D})$, sending each object $A \in \mathcal{C}$ to an object $F(A) \in \mathcal{D}$,
2. for each pair of objects $A, B \in \mathcal{C}$, a function $F_{A,B} : \mathcal{C}(A, B) \rightarrow \mathcal{D}(F(A), F(B))$, sending each morphism $f : A \rightarrow B$ to a morphism $F(f) : F(A) \rightarrow F(B)$,

satisfying:

1. (*Preservation of composition*) $F(g \circ f) = F(g) \circ F(f)$ for all composable f, g ,
2. (*Preservation of identities*) $F(\text{id}_A) = \text{id}_{F(A)}$ for all A .

A functor thus sends objects to objects and morphisms to morphisms, in a way that respects the categorical structure. Composition is preserved: composing two morphisms and then applying F gives the same result as applying F to each and then composing. Identities are preserved: the identity at A is sent to the identity at $F(A)$.

A *contravariant functor* $F : \mathcal{C} \rightarrow \mathcal{D}$ reverses the direction of morphisms: it sends $f : A \rightarrow B$ to $F(f) : F(B) \rightarrow F(A)$, and the composition rule becomes $F(g \circ f) = F(f) \circ F(g)$. A contravariant functor from $\mathcal{C}$ is the same as a (covariant) functor from $\mathcal{C}^{\text{op}}$.

The most important functors for deep learning are the following.

The *forgetful functor* $U : \mathbf{FVect}_{\mathbb{R}} \rightarrow \mathbf{Set}$ sends a finite-dimensional vector space to its underlying set, forgetting the linear structure. This captures the idea that a vector space is, in particular, a set. Similarly, one might forget the smooth structure of a manifold to obtain a bare topological space, or forget the topological structure to obtain a set. Forgetful functors always have left adjoints (free constructions), a fact with deep implications.

The *free vector space functor* $F : \mathbf{Set} \rightarrow \mathbf{FVect}_{\mathbb{R}}$ sends a finite set S to the vector space $\mathbb{R}^S$ with basis S . A function $f : S \rightarrow T$ of sets is sent to the linear map that extends f linearly: the basis element e_s maps to $e_{f(s)}$. One-hot encoding in machine learning is precisely the counit of the adjunction between F and U : it embeds each element of a finite set as the corresponding basis vector in the free vector space.

The *tangent functor* $T : \mathbf{Smooth} \rightarrow \mathbf{Smooth}$ sends a smooth manifold M to its tangent bundle TM and a smooth map $f : M \rightarrow N$ to its derivative (pushforward) $Tf = df : TM \rightarrow TN$. This is the categorical expression of the chain rule: the derivative of a composition $g \circ f$ is $d(g \circ f) = dg \circ df$, which is exactly the functoriality condition $T(g \circ f) = T(g) \circ T(f)$.

This last example is central. The chain rule — the computational backbone of backpropagation — is a theorem about the tangent functor, a consequence of its functoriality. This is not merely a restatement; it is the beginning of an explanation. The chain rule holds not because of any special property of real-valued functions, but because it must hold for any functor: functoriality is the conceptual source of the chain rule, and any reverse-mode differentiation algorithm that is correct must, at some level, be implementing a functorial construction.

1.4 Natural Transformations: Maps Between Functors

Given two functors $F, G : \mathcal{C} \rightarrow \mathcal{D}$, a *natural transformation* $\eta : F \Rightarrow G$ is a coherent family of morphisms in $\mathcal{D}$, one for each object of $\mathcal{C}$, that together express a “systematic” relationship between F and G .

Definition 1.3 (Natural Transformation). Let $F, G : \mathcal{C} \rightarrow \mathcal{D}$ be functors. A *natural transformation* $\eta : F \Rightarrow G$ is a family of morphisms $\{\eta_A : F(A) \rightarrow G(A)\}_{A \in \text{ob}(\mathcal{C})}$, called the *components* of η , such that for every morphism $f : A \rightarrow B$ in $\mathcal{C}$, the following square commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ \eta_A \downarrow & & \downarrow \eta_B \\ G(A) & \xrightarrow{G(f)} & G(B) \end{array}$$

This commutativity condition is the *naturality square* for f .

The naturality condition is the key. It says that η does not merely assign a morphism $\eta_A : F(A) \rightarrow G(A)$ to each object, but does so in a way that is *compatible with all the morphisms in $\mathcal{C}$* . The transformation from F to G commutes with the action of every morphism.

Natural transformations compose: if $\eta : F \Rightarrow G$ and $\mu : G \Rightarrow H$, then $\mu \circ \eta : F \Rightarrow H$ is defined componentwise by $(\mu \circ \eta)_A = \mu_A \circ \eta_A$. With this composition, the collection of all functors from $\mathcal{C}$ to $\mathcal{D}$ and all natural transformations between them forms a category, the *functor category* $[\mathcal{C}, \mathcal{D}]$ (also written $\mathcal{D}^{\mathcal{C}}$).

A natural transformation $\eta : F \Rightarrow G$ is a *natural isomorphism* if every component η_A is an

isomorphism in $\mathcal{D}$. In this case $G \cong F$ as functors, and we write $F \cong G$.

Let us see where natural transformations appear in machine learning. Consider two architectures for processing sequential data: an LSTM and an attention-based Transformer. Both can be formalised as functors from some category of input sequences to a category of hidden-state sequences. A “translation” between them — an assignment that maps every LSTM computation to a Transformer computation in a way that is compatible with the compositional structure of how sequences are processed — would be a natural transformation. Whether such translations exist, and what properties they must have, is a well-posed categorical question. In practice, the search for architectural equivalences (e.g., the equivalence between linear attention and linear recurrences) is, at root, the search for natural isomorphisms between certain functors.

A more concrete example: the softmax function appears in the definition of attention. It can be viewed as a natural transformation $\sigma : \mathbb{R}^{(-)} \Rightarrow \Delta^{(-)}$ from the functor sending n to $\mathbb{R}^n$, to the functor sending n to the probability simplex $\Delta^{n-1} \subset \mathbb{R}^n$. The naturality condition here would require that softmax commutes with relabelling of indices — which it does, since softmax is permutation-equivariant. This equivariance is precisely naturality.

1.5 Isomorphisms, Equivalences, and Adjunctions

A morphism $f : A \rightarrow B$ in a category is an *isomorphism* if there exists $g : B \rightarrow A$ such that $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$. We then write $A \cong B$.

A functor $F : \mathcal{C} \rightarrow \mathcal{D}$ is an *equivalence of categories* if there exists a functor $G : \mathcal{D} \rightarrow \mathcal{C}$ and natural isomorphisms $\eta : \text{Id}_{\mathcal{C}} \cong G \circ F$ and $\varepsilon : F \circ G \cong \text{Id}_{\mathcal{D}}$. The notion of equivalence is weaker than isomorphism of categories (which would require $G \circ F = \text{Id}_{\mathcal{C}}$ on the nose), but it is the correct notion in practice. Two equivalent categories are, for all categorical purposes, indistinguishable.

Adjunctions are the engine of category theory: most fundamental constructions — products, exponentials, free algebras, limits, colimits — arise as adjoint functors.

Definition 1.4 (Adjunction). An *adjunction* $F \dashv G$, with $F : \mathcal{C} \rightarrow \mathcal{D}$ the *left adjoint* and $G : \mathcal{D} \rightarrow \mathcal{C}$ the *right adjoint*, is a natural isomorphism of hom-sets:

$$\mathcal{D}(F(A), B) \cong \mathcal{C}(A, G(B))$$

natural in both $A \in \mathcal{C}$ and $B \in \mathcal{D}$.

Equivalently, an adjunction is given by the *unit* $\eta : \text{Id}_{\mathcal{C}} \Rightarrow G \circ F$ and *counit* $\varepsilon : F \circ G \Rightarrow \text{Id}_{\mathcal{D}}$, natural transformations satisfying the *triangle identities*: $(\varepsilon_F) \circ (F\eta) = \text{id}_F$ and $(G\varepsilon) \circ (\eta_G) = \text{id}_G$.

The free-forgetful adjunction $F \dashv U$ between **Set** and **FVect** $_{\mathbb{R}}$ is the canonical example. The unit $\eta_S : S \rightarrow U(F(S))$ is the inclusion of a set S into its free vector space (the one-hot encoding); the counit $\varepsilon_V : F(U(V)) \rightarrow V$ is the linear extension of the identity map on a basis, the unique linear map sending each basis vector to itself.

Adjunctions encode a fundamental asymmetry: the left adjoint F is “free” or “forgetful of constraints”, while the right adjoint G is “cofree” or “enriched with structure”. In deep learning, the relationship between the forward pass (which applies a learned map) and the backward pass (which computes its adjoint, in a sense we will make precise) is adjunctional in a deep way.

Chapter 2

Monoidal Categories and String Diagrams

2.1 Composing Systems in Parallel

In a plain category, we can compose systems in *series*: given $f : A \rightarrow B$ and $g : B \rightarrow C$, we get $g \circ f : A \rightarrow C$. But many systems can also be composed in *parallel*: given two independent subsystems, we want to run them side by side, processing independent inputs and producing independent outputs. The categorical structure that captures this is a *monoidal category*.

Definition 2.1 (Monoidal Category). A *monoidal category* $(\mathcal{C}, \otimes, I, \alpha, \lambda, \rho)$ is a category $\mathcal{C}$ equipped with:

- a *tensor product* bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$,
- a *unit object* $I \in \text{ob}(\mathcal{C})$,
- natural isomorphisms:

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C) \quad (\text{associator}),$$

$$\lambda_A : I \otimes A \xrightarrow{\sim} A \quad (\text{left unitor}),$$

$$\rho_A : A \otimes I \xrightarrow{\sim} A \quad (\text{right unitor}),$$

satisfying the *pentagon* and *triangle* coherence equations.

The pentagon equation governs the associator: four different ways to re-parenthesise a fourfold tensor product all agree. The triangle equation governs the interaction of the associator with the unitors. Mac Lane's Coherence Theorem then tells us that *any* diagram built from α , λ , ρ and their inverses commutes automatically — we may re-parenthesise freely without tracking which associativity isomorphisms we applied, as long as we apply the correct number.

The archetypal monoidal categories for deep learning are the following.

In $\mathbf{FVect}_{\mathbb{R}}$, the tensor product is the usual tensor product of vector spaces: $\mathbb{R}^m \otimes \mathbb{R}^n \cong \mathbb{R}^{mn}$, with unit object $\mathbb{R}^1 \cong \mathbb{R}$. The tensor product of two linear maps $f : V \rightarrow V'$ and $g : W \rightarrow W'$ is the linear map $f \otimes g : V \otimes W \rightarrow V' \otimes W'$ that acts as $(f \otimes g)(v \otimes w) = f(v) \otimes g(w)$, extended linearly. In terms of matrices, $f \otimes g$ is the Kronecker product of the corresponding matrices.

In $\mathbf{Set}$, the cartesian product $\times$ gives a monoidal structure (in fact a *cartesian* monoidal structure, as we discuss below), with unit the singleton set $\{*\}$. The product of two functions $f : A \rightarrow B$ and $g : C \rightarrow D$ is $f \times g : A \times C \rightarrow B \times D$, $(a, c) \mapsto (f(a), g(c))$.

In the category $\mathbf{Stoch}$ of finite sets and stochastic matrices (Markov kernels), the tensor product is again the cartesian product on objects and the independent coupling of stochastic kernels on morphisms: if $f : A \rightarrow \mathcal{P}(B)$ and $g : C \rightarrow \mathcal{P}(D)$ are stochastic maps (sending each element to a probability distribution over the codomain), then $f \otimes g : A \times C \rightarrow \mathcal{P}(B \times D)$ sends (a, c) to the product measure $f(a) \otimes g(c)$. This captures the notion of two independent stochastic processes.

2.2 Cartesian and Compact Closed Categories

Among monoidal categories, two special cases are of paramount importance.

A monoidal category is *cartesian* if $\otimes = \times$ is the categorical product and I is the terminal object. In a cartesian category, every object A carries natural diagonal and projection maps:

$$\Delta_A : A \rightarrow A \times A, \quad \pi_1 : A \times B \rightarrow A, \quad \pi_2 : A \times B \rightarrow B.$$

These express the ability to *copy* and *discard* information, and they are fundamental to the semantics of classical computation: a variable in a program can be used multiple times (copying) and can go out of scope (discarding). The category $\mathbf{Set}$ is cartesian, as is the category of C^∞ -manifolds with smooth maps, and the category of measurable spaces with measurable functions.

A monoidal category is *compact closed* (or a *compact category*) if every object A has a *dual* A^* , together with morphisms:

$$\eta_A : I \rightarrow A^* \otimes A \quad (\text{unit or coevaluation}), \quad \varepsilon_A : A \otimes A^* \rightarrow I \quad (\text{counit or evaluation}),$$

satisfying the *snake equations*:

$$(\varepsilon_A \otimes \text{id}_A) \circ (\text{id}_A \otimes \eta_A) = \text{id}_A, \quad (\text{id}_{A^*} \otimes \varepsilon_A) \circ (\eta_A \otimes \text{id}_{A^*}) = \text{id}_{A^*}.$$

The category $\mathbf{FVect}_{\mathbb{R}}$ is compact closed: the dual of $\mathbb{R}^n$ is $(\mathbb{R}^n)^* \cong \mathbb{R}^n$ (the space of linear functionals), the unit is the map $\mathbb{R} \rightarrow \mathbb{R}^n \otimes \mathbb{R}^n$ given by $1 \mapsto \sum_i e_i \otimes e_i$, and the counit is the trace pairing $e_i \otimes e_j \mapsto \delta_{ij}$. Compact closure gives an intrinsic notion of transposition: the transpose of a linear map $f : A \rightarrow B$ is $f^* = (\text{id}_{A^*} \otimes \varepsilon_B) \circ (\text{id}_{A^*} \otimes f \otimes \text{id}_{B^*}) \circ (\eta_A \otimes \text{id}_{B^*}) : B^* \rightarrow A^*$. In matrix terms this is the literal matrix transpose.

The fact that $\mathbf{FVect}_{\mathbb{R}}$ is compact closed, while $\mathbf{Set}$ is cartesian but not compact (you

cannot take the “dual” of a set in any meaningful way), marks a deep difference between linear and nonlinear computation. Neural networks live primarily in $\mathbf{FVect}_{\mathbb{R}}$ for their parameter spaces, but their input-output behaviour may involve nonlinear (hence cartesian, not compact) structure.

2.3 String Diagrams

String diagrams are a graphical calculus for monoidal categories that are both a convenience and a conceptual tool. The key theorem, due to Joyal and Street, is:

Theorem 2.2 (Joyal–Street Coherence for String Diagrams). *In a (symmetric) monoidal category, a (symmetric) monoidal equation between composite morphisms holds if and only if the corresponding string diagrams are equal as planar (resp. topological) isotopy classes of diagrams.*

The rules are as follows. Objects (types) are represented as labelled wires running from top to bottom. Morphisms are represented as boxes with input wires at the top and output wires at the bottom. Sequential composition $g \circ f$ corresponds to placing f above g (wires from f feed into g). Tensor product $f \otimes g$ corresponds to placing f and g side by side. The tensor unit I is represented by an empty wire (no wire). Identity morphisms are straight wires.

In $\mathbf{FVect}_{\mathbb{R}}$, a string diagram is essentially a tensor network. Each wire carries a vector space label (a tensor index), and each box is a tensor (a multi-linear map). Contraction of indices corresponds to connecting an output wire to an input wire (composition). The graphical notation used in quantum computing for circuit diagrams and in physics for Feynman diagrams are both instances of string diagrams in specific monoidal categories.

For deep learning, the key insight is that a *feedforward neural network* is precisely a string diagram in an appropriate monoidal category of smooth or piecewise-smooth maps. Specifically, consider the category **Smooth** with monoidal structure given by cartesian product. A neural network layer $f_W : \mathbb{R}^m \rightarrow \mathbb{R}^n$ (where W are the weights) is a morphism. The full network $f_{W_n} \circ \cdots \circ f_{W_1}$ is obtained by sequential composition, and a network that processes two independent inputs in parallel (e.g., a two-branch architecture) is obtained by tensor product. String diagrams make the topology of information flow explicit and provably equal to the underlying categorical expression.

The real power of string diagrams appears when we move to more structured monoidal categories. In a compact closed category, we can bend wires: a wire for A^* can be thought of as a wire for A running “upward” (backward in time). The snake equations become the statement that a wire bent upward and then back down is the same as a straight wire. This bending is what, in the categorical treatment of backpropagation, will correspond to the reversal of direction from the forward to the backward pass.

2.4 Neural Networks as Morphisms in a Monoidal Category

Fong, Spivak, and Tuyéras (2019) gave the first systematic categorical formulation of supervised learning. They worked in a category they called **Learn**, whose objects are Euclidean spaces and whose morphisms are *learners*: tuples (P, I, U, r) where P is a parameter space, $I : P \times A \rightarrow B$ is the *implementation* function (the forward pass), $U : P \times A \times B \rightarrow P$ is the *update* function (the gradient step that updates parameters given an input and a target), and $r : P \times A \times B \rightarrow A$ is the *request* function (the backward pass that propagates the “error signal” to the input).

The composition of two learners $f = (P, I_f, U_f, r_f) : A \rightarrow B$ and $g = (Q, I_g, U_g, r_g) : B \rightarrow C$ is the learner $g \circ f : A \rightarrow C$ with parameter space $P \times Q$, implementation $I_{g \circ f}((p, q), a) = I_g(q, I_f(p, a))$, and backward computations that propagate signals in the appropriate order. The key result is that this composition is associative (up to isomorphism of parameter spaces), and that **Learn** is a symmetric monoidal category, with tensor product given by independent processing of independent inputs.

This framework immediately raises the question: what is the precise relationship between **Learn** and other categorical structures? The answer comes through the lens construction, to which we turn in the next chapter, and which reveals that **Learn** is a subcategory of a more fundamental categorical object.

Chapter 3

Lenses, Para, and Backpropagation as a Functor

3.1 The Lens Pattern

The concept of a *lens* originated in functional programming as an abstraction for the notion of a “field accessor”: a pair of functions that can read and write a subpart of a larger data structure. In the pure categorical setting, they emerge as the morphisms of a category that captures bidirectional computation.

Definition 3.1 (Lens). Given a cartesian category $(\mathcal{C}, \times, 1)$, a *lens* from (S, A) to (T, B) is a pair of morphisms $(\mathbf{get}, \mathbf{put})$ where:

$$\mathbf{get} : S \rightarrow A \quad (\text{the forward pass, or “getter”}),$$

$$\mathbf{put} : S \times B \rightarrow T \quad (\text{the backward pass, or “putter”}).$$

In the classical functional programming setting, one takes $S = T$ and $A = B$, giving a lens $(S, A) \rightarrow (S, A)$ — a way to read and write a value of type A within a container of type S . The *lens laws* (well-behavedness conditions) require: (1) $\mathbf{get} \circ \mathbf{put}(s, a) = a$ (you get back what you put in); (2) $\mathbf{put}(s, \mathbf{get}(s)) = s$ (putting what you already read leaves things unchanged); (3) $\mathbf{put}(\mathbf{put}(s, a), a') = \mathbf{put}(s, a')$ (the most recent put wins). These three laws correspond exactly to the unit and associativity conditions of certain algebraic structures.

For deep learning, the relevant generalisation is the *simple lens* or *bimorphic lens*, where $S \neq T$ and $A \neq B$ are allowed. Given a smooth function $f : X \rightarrow Y$ and a loss function $\ell : Y \rightarrow \mathbb{R}$ (which is itself a morphism), the pair (f, f') where $f' : X \times T^*Y \rightarrow T^*X$ sends $(x, \delta y)$ to the pullback of δy along f is a lens in the tangent bundle category. The getter f computes the forward pass, and the putter f' propagates gradient information backward, which is exactly the operation required in the backward pass of backpropagation.

3.2 Composition of Lenses

The critical property of lenses is that they *compose*: given lenses $\ell_1 : (S_1, A_1) \rightarrow (S_2, A_2)$ and $\ell_2 : (S_2, A_2) \rightarrow (S_3, A_3)$ with components $(\mathbf{get}_1, \mathbf{put}_1)$ and $(\mathbf{get}_2, \mathbf{put}_2)$, the composite lens $\ell_2 \circ \ell_1 : (S_1, A_1) \rightarrow (S_3, A_3)$ has:

$$\begin{aligned}\mathbf{get}_{\ell_2 \circ \ell_1} &= \mathbf{get}_2 \circ \mathbf{get}_1, \\ \mathbf{put}_{\ell_2 \circ \ell_1}(s_1, \delta_3) &= \mathbf{put}_1(s_1, \mathbf{put}_2(\mathbf{get}_1(s_1), \delta_3)).\end{aligned}$$

The composition formula for **put** encodes precisely the structure of the backpropagation algorithm. To propagate the gradient δ_3 from the output of the composed system back to the input of the first subsystem, we first need to know the intermediate state $\mathbf{get}_1(s_1)$ (the activation at the boundary between the two subsystems, which is computed during the forward pass and cached), and then apply $\mathbf{put}_2$ to propagate δ_3 backward through the second subsystem, obtaining δ_2 , and then apply $\mathbf{put}_1$ to propagate δ_2 backward through the first. This is exactly what backpropagation computes, and the composition formula makes it explicit that the sequential structure of the backward pass mirrors the sequential structure of the forward pass, but in reverse order.

This compositionality is the key theorem: lenses form a category. Let us state it precisely.

Definition 3.2 (Lens Category). Let $\mathcal{C}$ be a cartesian category. The *lens category* $\mathbf{Lens}(\mathcal{C})$ has:

- objects: pairs (S, A) with $S, A \in \text{ob}(\mathcal{C})$,
- morphisms: lenses $(\mathbf{get}, \mathbf{put}) : (S, A) \rightarrow (T, B)$ with $\mathbf{get} : S \rightarrow B$ and $\mathbf{put} : S \times A \rightarrow T$,
- composition as defined above,
- identity lens: $\text{id}_{(S,A)} = (\text{id}_S, \pi_1) : (S, A) \rightarrow (S, A)$ with $\mathbf{get} = \text{id}_S$ and $\mathbf{put}(s, a) = s$.

Proposition 3.3. $\mathbf{Lens}(\mathcal{C})$ is a well-defined category. Composition is associative and respects identity lenses.

Proof. Associativity of composition of getters follows from associativity of composition in $\mathcal{C}$. For putters: given three composable lenses ℓ_1, ℓ_2, ℓ_3 with getters g_1, g_2, g_3 and putters p_1, p_2, p_3 , both $(\ell_3 \circ \ell_2) \circ \ell_1$ and $\ell_3 \circ (\ell_2 \circ \ell_1)$ compute the same backward function: $p_1(s_1, p_2(g_1(s_1), p_3(g_2(g_1(s_1)), \delta_4)))$, since function composition is associative. The unit laws follow directly from the definition of the identity lens. $\square$

The lens category has a monoidal structure as well: given lenses $\ell_1 : (S_1, A_1) \rightarrow (T_1, B_1)$ and $\ell_2 : (S_2, A_2) \rightarrow (T_2, B_2)$, their tensor product is the lens $(S_1 \times S_2, A_1 \times A_2) \rightarrow (T_1 \times T_2, B_1 \times B_2)$ that applies ℓ_1 to the first component and ℓ_2 to the second independently. This corresponds to processing two independent subcomputations in parallel without any interaction between their forward or backward passes.

3.3 The Para Construction

The lens category describes the bidirectional flow of information but does not yet include the *learnable parameters*. Parameters enter through the *Para construction*, which equips every morphism with a “knob” that can be tuned.

Definition 3.4 (Para Construction). Let $(\mathcal{C}, \otimes, I)$ be a monoidal category. The *Para category* $\mathbf{Para}(\mathcal{C})$ has:

- the same objects as $\mathcal{C}$,
- a morphism $A \rightarrow B$ in $\mathbf{Para}(\mathcal{C})$ is a pair (P, f) where $P \in \text{ob}(\mathcal{C})$ is a parameter object and $f : P \otimes A \rightarrow B$ is a morphism in $\mathcal{C}$,
- two morphisms (P, f) and $(Q, g) : A \rightarrow B$ are considered equal when they are related by a reparametrisation: there exists an isomorphism $\phi : P \rightarrow Q$ such that $g \circ (\phi \otimes \text{id}_A) = f$,
- the composition of $(P, f) : A \rightarrow B$ and $(Q, g) : B \rightarrow C$ is $(P \otimes Q, h) : A \rightarrow C$, where $h = g \circ (\text{id}_Q \otimes f) : Q \otimes P \otimes A \rightarrow C$ (first apply f using parameters P , then apply g using parameters Q).

A feedforward neural network is precisely a morphism in $\mathbf{Para}(\mathbf{Smooth})$ (or $\mathbf{Para}(\mathbf{FVect})$ for linear networks). A network with parameter spaces $P_1, P_2, \dots, P_n$ for its n layers is a morphism with total parameter object $P_1 \otimes P_2 \otimes \dots \otimes P_n$ and implementation function $f_{P_n} \circ \dots \circ f_{P_1}$. The Para construction captures the idea that a parametrised family of functions is a single morphism in an appropriate category.

The key categorical insight is now the following. The gradient descent update step for parameters is a morphism in $\mathbf{Para}(\mathcal{C})$ itself: the update function $U : P \times A \times B \rightarrow P$ (given current parameters p , input a , and target b , produce updated parameters p') is a morphism. The learning algorithm is not merely an operation *on* the network but is *part of the categorical structure* of the Para category.

3.4 Backpropagation as a Functor: The Cruttwell–Gavranović–Ghani–Wilson Theorem

The key theorem of the categorical treatment of backpropagation, proved by Cruttwell, Gavranović, Ghani, and Wilson (2022), establishes that backpropagation is not merely an algorithm but a *functor*: a structure-preserving map between categories that sends a computational graph (a morphism in a suitable Para category) to the corresponding gradient computation.

We work in the category $\mathbf{Smooth}$ of smooth manifolds and smooth maps. The *reverse derivative category* structure on $\mathbf{Smooth}$, developed by Cockett and Cruttwell through the theory of *cartesian reverse differential categories*, gives the abstract framework; we describe it concretely here.

Definition 3.5 (Cartesian Reverse Differential Category). A cartesian category $(\mathcal{C}, \times, 1)$ is a *cartesian reverse differential category* (CRDC) if it is equipped with an operator R that

assigns to each morphism $f : A \rightarrow B$ a morphism $R[f] : A \times B \rightarrow A$ (the *reverse derivative*, or *gradient*, of f), subject to the following axioms (writing $f : A \rightarrow B$, $g : B \rightarrow C$, $\pi_1 : A \times B \rightarrow A$, $\pi_2 : A \times B \rightarrow B$):

1. *Linearity in the second argument*: $R[f](a, -) : B \rightarrow A$ is a linear map in the appropriate sense,
2. *Additivity*: $R[f + g] = R[f] + R[g]$ (for additive structure on $\text{hom}(A, B)$),
3. *Identity*: $R[\text{id}_A](a, a') = a'$,
4. *Chain rule*: $R[g \circ f](a, c') = R[f](a, R[g](f(a), c'))$,
5. *Projection*: $R[\pi_1](a \times b, a') = (a', 0)$.

Axiom 4 is the chain rule in categorical form: to compute the reverse derivative of $g \circ f$ at input a with output cotangent c' , we first evaluate f at a (the forward pass), propagate c' backward through g using $R[g]$, and then propagate the result backward through f using $R[f]$. This is precisely the structure of the backward pass.

The category **Smooth** with $R[f](x, \delta y) = (Df(x))^\top \delta y$ (the transpose of the Jacobian applied to the output cotangent) is a CRDC. The chain rule axiom corresponds to the classical chain rule for Jacobians: $(D(g \circ f)(x))^\top = (Df(x))^\top (Dg(f(x)))^\top$.

The main theorem can now be stated as follows.

Theorem 3.6 (Cruttwell–Gavranović–Ghani–Wilson, 2022). *There exists a functor*

$$\text{BP} : \mathbf{Para}(\mathbf{Smooth}) \rightarrow \mathbf{Lens}(\mathbf{Smooth})$$

sending each parametrised smooth map $f : P \times A \rightarrow B$ to the lens $(\text{get}_f, \text{put}_f) : (P \times A, B) \rightarrow (P \times A, B)$ where:

$$\begin{aligned} \text{get}_f(p, a) &= f(p, a), \\ \text{put}_f((p, a), \delta b) &= R[f]((p, a), \delta b) = (D_{(p,a)}f)^\top \delta b. \end{aligned}$$

*This functor is a symmetric monoidal functor, sending tensor products in **Para(Smooth)** (parallel composition of independent parametrised maps) to tensor products in **Lens(Smooth)** (parallel composition of independent bidirectional processes).*

The proof of functoriality reduces to the chain rule: $\text{BP}(g \circ f) = \text{BP}(g) \circ \text{BP}(f)$ is exactly the statement that the backward pass of a composed network equals the composition of the backward passes in reverse order, which follows from the CRDC chain rule axiom applied twice.

What makes this theorem significant is not that it tells us something we did not know about backpropagation — the algorithm was clear before — but that it identifies the precise categorical structure that *explains* why backpropagation is correct and compositional. The functoriality of BP guarantees that any modular decomposition of a network into layers corresponds to a modular decomposition of the backward pass, and that these decompositions are compatible.

Chapter 4

Optics, Tambara Modules, and Automatic Differentiation

4.1 Beyond Lenses: The Need for Optics

Lenses, as defined in the previous chapter, are adequate for describing bidirectional processes in *cartesian* categories, where copying and discarding of data are available. But many important structures in deep learning live in categories that are monoidal but not cartesian: the category $\mathbf{FVect}_{\mathbb{R}}$ of vector spaces and linear maps is not cartesian (there is no natural diagonal map $V \rightarrow V \otimes V$ that is linear), and many probabilistic categories (where morphisms are Markov kernels) are similarly non-cartesian. For these settings, the notion of a lens must be generalised to that of an *optic*.

The theory of optics, developed by Milewski, Riley, Boisseau, and others in the context of functional programming and later brought to full mathematical generality by Boisseau and Gibbons, Clarke, Elkins, and others, provides such a generalisation. The key idea is to replace the explicit get-put pair of a lens by a single morphism that lives in a certain coend, capturing the “residual state” that must be passed from the forward to the backward pass without specifying its type concretely.

Definition 4.1 (Optic). Let $(\mathcal{C}, \otimes, I)$ be a monoidal category. An *optic* from (S, T) to (A, B) is an element of the coend:

$$\mathbf{Optic}_{\mathcal{C}}((S, T), (A, B)) = \int^{M \in \mathcal{C}} \mathcal{C}(S, M \otimes A) \times \mathcal{C}(M \otimes B, T).$$

Let us unpack this definition. A coend $\int^M F(M, M)$ over a functor $F : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set}$ is a generalised notion of “quotient” or “existential type” over M : it is the set of pairs (m, n) where $m \in F(M, M)$ for some object M , quotiented by the relation that identifies (m, n) with $(F(h, \text{id})(m), F(\text{id}, h)(n'))$ for any morphism $h : M \rightarrow M'$. In the case of optics, the coend identifies two representations of the same bidirectional process that happen to factorise through different “residual” objects M but are naturally isomorphic.

Concretely: an optic from (S, T) to (A, B) is an equivalence class of triples (M, l, r) where

M is a *residual object* (the state passed from forward to backward), $l : S \rightarrow M \otimes A$ is the *forward* component (which splits the input S into a residual M and an “inner input” A), and $r : M \otimes B \rightarrow T$ is the *backward* component (which combines the residual M with an “inner output” B to produce T). Two such triples (M, l, r) and (M', l', r') are equivalent if there is a morphism $h : M \rightarrow M'$ such that $l' = (\text{id}_A \otimes h) \circ l$ and $r = r' \circ (h \otimes \text{id}_B)$.

In the cartesian case, optics reduce to lenses: taking $M = S$ and $l = \Delta_S = (s \mapsto (s, s))$ (the diagonal, using copy), we get a lens with $\text{get} = \pi_A \circ l$ and $\text{put}(s, b) = r(s, b)$. But in the non-cartesian case, the optic formulation allows the residual state to be an object that cannot be defined using copying.

4.2 Optics for Reverse-Mode Automatic Differentiation

The application of optics to reverse-mode automatic differentiation (AD) was developed by Elliot, Pearlmutter, and Siskind in the functional programming tradition, and given a precise categorical treatment by Cruttwell et al. and by Nunes and Gonçalves.

In the setting of $\mathbf{FVect}_{\mathbb{R}}$, consider a differentiable function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$. Its derivative at a point $x \in \mathbb{R}^m$ is a linear map $Df(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$, represented by the Jacobian matrix $J_f(x) \in \mathbb{R}^{n \times m}$. In reverse-mode AD, we do not compute $Df(x)$ explicitly; instead, we compute its adjoint (transpose) applied to a cotangent vector $\delta y \in \mathbb{R}^n$: the *vector-Jacobian product* $\delta x = J_f(x)^\top \delta y \in \mathbb{R}^m$.

The optic formulation makes this structure transparent. Working in $\mathbf{FVect}_{\mathbb{R}}$ (which is not cartesian, so lenses are not directly available), the optic from $(\mathbb{R}^m, \mathbb{R}^m)$ to $(\mathbb{R}^n, \mathbb{R}^n)$ corresponding to f is:

$$(\mathbb{R}^m, l = x \mapsto (x, f(x)), r = (x, \delta y) \mapsto J_f(x)^\top \delta y).$$

Here the residual object is $M = \mathbb{R}^m$ itself (the input, which is “saved” from the forward pass to be used in the backward pass), l is the function that computes both the intermediate activations and the output, and r is the function that applies the transpose Jacobian to a cotangent.

The composition of optics corresponds to the composition of the forward and backward passes. Given optics (M_1, l_1, r_1) from (A, A) to (B, B) and (M_2, l_2, r_2) from (B, B) to (C, C) , their composite from (A, A) to (C, C) is:

$$(M_1 \otimes M_2, l = l_2 \circ l_1, r = r_1 \circ (\text{id}_{M_1} \otimes r_2)).$$

The forward component of the composite first runs l_1 (computing intermediate activations for the first function and passing the output to the second), then runs l_2 ; the backward component first runs r_2 (propagating the cotangent from the output of the composite back to the boundary between the two functions) and then runs r_1 (propagating it back to the input). The residual state $M_1 \otimes M_2$ records the intermediate activations of both functions.

The correctness of reverse-mode AD — the guarantee that the computed cotangent δx is indeed $J_f(x)^\top \delta y$ for the composite function $f = f_n \circ \dots \circ f_1$ — follows from the functoriality

of the optic composition and the chain rule, which is now a consequence of the coend construction rather than an additional axiom.

4.3 Tambara Modules and the Abstract Theory of Optics

The deepest understanding of optics comes from their relationship to *Tambara modules*, which provide a representation theorem that generalises the Yoneda lemma.

Definition 4.2 (Tambara Module). Let $(\mathcal{C}, \otimes, I)$ be a monoidal category. A *Tambara module* (or *$\mathcal{C}$ -actegory profunctor*) over $\mathcal{C}$ is a profunctor $P : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set}$ equipped with natural transformations:

$$\alpha_{A,S,T} : P(S, T) \rightarrow P(A \otimes S, A \otimes T)$$

for each $A \in \mathcal{C}$, satisfying coherence conditions with respect to the monoidal structure of $\mathcal{C}$.

The coherence conditions require that α is compatible with the associator and unit of $\mathcal{C}$: $\alpha_{I,S,T} = \text{id}$ and $\alpha_{A \otimes B, S, T} = \alpha_{A, B \otimes S, B \otimes T} \circ \alpha_{B, S, T}$. These express the idea that the action of the monoidal structure on profunctors is itself monoidal.

The key representation theorem (Boisseau–Gibbons, Riley) is:

Theorem 4.3 (Tambara Representation of Optics). *There is a natural bijection:*

$$\mathbf{Optic}_{\mathcal{C}}((S, T), (A, B)) \cong \int_{P \in \mathbf{Tamb}(\mathcal{C})} \mathbf{Set}(P(A, B), P(S, T)),$$

where the end $\int_P$ ranges over all Tambara modules P and $\mathbf{Tamb}(\mathcal{C})$ is the category of Tambara modules.

In other words, an optic from (S, T) to (A, B) is the same as a natural transformation from $P(A, B)$ to $P(S, T)$, for all Tambara modules P simultaneously. This is a generalisation of the Yoneda lemma (which represents morphisms as natural transformations of hom-functors) to the bidirectional, mixed-variance setting.

The practical import for deep learning is as follows: any “lens-like” structure on a function space — any way of associating forward and backward computations that respects the monoidal structure of composition — must factor through the optic representation. This universality property means that optics are not one of many possible frameworks for bidirectional computation, but the *canonical* one: they are what every reasonable bidirectional structure turns out to be.

4.4 Mixed-Variance Functors and the Bidirectionality of Attention

A striking application of the optic framework is to the attention mechanism of the Transformer. Recall that scaled dot-product attention computes:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V,$$

where $Q, K, V \in \mathbb{R}^{n \times d}$ are the query, key, and value matrices (for a sequence of n tokens of dimension d). The attention weights $A = \text{softmax}(QK^\top / \sqrt{d_k})$ depend on both Q and K , and the output depends on both A and V .

From the optic perspective, attention is a morphism in a category of sequences with a non-trivial monoidal structure. The query-key interaction $QK^\top$ is a bilinear form — a morphism $\mathbb{R}^d \otimes \mathbb{R}^d \rightarrow \mathbb{R}$ in $\mathbf{FVect}_{\mathbb{R}}$. The softmax is a nonlinear operation that introduces a global normalisation across positions (a feature that cannot be expressed within a purely linear or Markovian framework). The output AV is a linear combination of value vectors weighted by attention scores.

The forward pass of attention has residual state (Q, K, A) : during the backward pass, computing the gradient with respect to Q requires K and the upstream gradient; computing the gradient with respect to K requires Q and the upstream gradient; computing the gradient with respect to V requires A and the upstream gradient. The optic structure makes explicit that this residual state has the form $M = \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$, and the backward component r is the function that computes all three input gradients simultaneously.

The categorical treatment also clarifies a subtlety: the gradient of the softmax requires the full attention matrix A , not just the attention scores before normalisation. This dependence is automatically captured by the optic structure, which requires the full residual state to be saved during the forward pass.

Chapter 5

Polynomial Functors and Neural Dynamical Systems

5.1 Polynomials as Wiring Diagrams

The category **Poly** of polynomial functors, developed by Spivak and his collaborators over the past decade, provides an unexpected framework for understanding recurrent neural networks and stateful computation. The connection is deep: every recurrent network can be described as a morphism in **Poly**, and the composition of recurrent networks corresponds to a specific monoidal product on **Poly** that Spivak calls the “substitution” product.

A *polynomial functor* is a functor $p : \mathbf{Set} \rightarrow \mathbf{Set}$ of the form:

$$p(Y) = \sum_{i \in p(1)} Y^{p[i]},$$

where $p(1)$ is the set of “positions” of p , and for each position $i \in p(1)$, $p[i]$ is the set of “directions” at i . In other words, a polynomial functor is determined by a set of positions together with, for each position, a set of directions (or inputs). We write $p = \sum_{i \in p(1)} \mathbf{y}^{p[i]}$, using the suggestive notation $\mathbf{y}^S$ for the representable functor $\mathbf{Set}(S, -)$.

The category **Poly** has polynomial functors as objects and natural transformations between them as morphisms. A morphism $\phi : p \rightarrow q$ in **Poly** consists of:

1. a function on positions $\phi_1 : p(1) \rightarrow q(1)$, sending each position of p to a position of q ,
2. for each position $i \in p(1)$, a function on directions $\phi_i^\# : q[\phi_1(i)] \rightarrow p[i]$, going *backwards* from the directions of q at $\phi_1(i)$ to the directions of p at i .

The backward direction of $\phi_i^\#$ is the hallmark of a polynomial morphism and is what distinguishes **Poly** from the category of mere sets-over-sets. It is precisely the contravariance in the directions that will encode the backward pass of a recurrent network.

5.2 Dynamical Systems in **Poly**

Spivak observes that a discrete-time dynamical system with state space S , output space O , and input space I can be described as a morphism in **Poly**:

$$\phi : S \cdot y^S \rightarrow O \cdot y^I,$$

where $S \cdot y^S$ is the polynomial with $|S|$ positions each having S directions (a “full” polynomial on S), and similarly $O \cdot y^I$ has $|O|$ positions each with I directions.

Concretely, ϕ consists of:

1. $\phi_1 : S \rightarrow O$, the *readout function* that maps each state to an output,
2. $\phi^\# : S \times I \rightarrow S$ (using the forward map from positions of the codomain to apply the directions), the *update function* or *transition function* of the dynamical system.

A recurrent neural network is precisely a dynamical system in this sense, with state space $S = \mathbb{R}^d$ (the hidden state dimension), input space $I = \mathbb{R}^n$ (the input dimension), and output space $O = \mathbb{R}^k$ (the output dimension). The update function $\phi^\# : \mathbb{R}^d \times \mathbb{R}^n \rightarrow \mathbb{R}^d$ is the RNN cell (e.g., $h_{t+1} = \tanh(W_h h_t + W_x x_t + b)$ for a vanilla RNN), and the readout $\phi_1 : \mathbb{R}^d \rightarrow \mathbb{R}^k$ is the output projection.

The power of the polynomial framework is that it describes not just individual dynamical systems but their *composition*. The *substitution product* $p \otimes q$ of two polynomials corresponds to putting one dynamical system inside another: the outputs of the inner system serve as inputs to the outer system, and the states of both are maintained simultaneously. Formally:

$$(p \otimes q)(Y) = p(q(Y)).$$

Under this product, the morphism for the composite dynamical system is the composite morphism $\phi \otimes \psi : p \otimes q \rightarrow r$ in **Poly**, and the resulting system has state space $S_p \times S_q$, update function determined by the updates of the two components and the coupling between them, and readout determined by the outer readout composed with the inner one.

5.3 The Mamba Architecture as a Polynomial Morphism

The selective state space model of the Mamba architecture (Gu and Dao, 2023) can be placed in the polynomial framework with interesting clarity. Recall that Mamba computes, for each input $x_t \in \mathbb{R}^d$:

$$\begin{aligned} \bar{A}_t &= \exp(\Delta_t A), & \bar{B}_t &= \Delta_t B(x_t), \\ h_t &= \bar{A}_t h_{t-1} + \bar{B}_t x_t, \\ y_t &= C(x_t) h_t, \end{aligned}$$

where $A \in \mathbb{R}^{d \times d}$ is a fixed diagonal matrix, $\Delta_t, B(x_t), C(x_t)$ are functions of the input, and $h_t \in \mathbb{R}^d$ is the hidden state. The content-dependent discretisation parameters Δ_t are what

makes Mamba “selective”: the effective state transition matrix $\bar{A}_t$ depends on the content of the current input, not just its position.

As a polynomial morphism, Mamba is $\phi : \mathbb{R}^d \cdot \mathbf{y}^{\mathbb{R}^d} \rightarrow \mathbb{R}^d \cdot \mathbf{y}^{\mathbb{R}^d}$ with:

$$\phi_1(h) = C(x)h \quad (\text{readout, depending on the current input } x),$$

$$\phi^\#(h, \delta h') = \bar{A}^\top \delta h' + \bar{B}^\top \delta y \quad (\text{backward update, for the gradient computation}).$$

The selectivity of Mamba corresponds, in the polynomial language, to the fact that the structural maps ϕ_1 and $\phi^\#$ both depend on the current input x , making them morphisms in a suitable enriched version of **Poly** rather than in the bare category.

5.4 Polynomial Coalgebras and Memory

A *coalgebra* for a polynomial functor p in **Poly** is an object S together with a morphism $c : S \rightarrow p(S)$. In the dynamical systems context, $c : S \rightarrow O \times S^I$ (for the polynomial $O \cdot \mathbf{y}^I$) consists of a readout $S \rightarrow O$ and an update $S \times I \rightarrow S$, which is exactly a discrete-time dynamical system on state space S .

The theory of coalgebras over a functor p is dual to the theory of algebras: where algebras are initial (the free algebra is built inductively from generators), coalgebras are terminal (the *final* coalgebra is the object of all infinite behaviours). For the polynomial $p = O \cdot \mathbf{y}^I$, the final coalgebra is the set of all functions $I^\omega \rightarrow O^\omega$ (from infinite input streams to infinite output streams) — the set of all possible behaviours of dynamical systems with input I and output O .

This connects to the theory of recurrent networks in an illuminating way. The expressivity question for RNNs — which input-output functions $f : I^\omega \rightarrow O^\omega$ can be approximated by an RNN with states in $\mathbb{R}^d$? — is the question of which elements of the final coalgebra can be represented by a polynomial coalgebra $c : \mathbb{R}^d \rightarrow O \cdot (\mathbb{R}^d)^I$. The universal approximation theorem for RNNs (under appropriate compactness assumptions) is the statement that the coalgebra maps from $\mathbb{R}^d$ are dense in the final coalgebra with the product topology.

Chapter 6

Categorical Probability and Markov Categories

6.1 The Problem of Probability in Category Theory

Probability theory has long resisted categorical treatment. The difficulty is that probability measures do not compose in the same clean way that functions compose. Given a probability distribution μ on X and a stochastic kernel $k : X \rightarrow \mathcal{P}(Y)$ (a measurable function assigning a probability measure on Y to each point in X), the composition is the pushforward measure $k_*\mu = \int_X k(x, -) d\mu(x)$, which involves an integration. This integration is not as elementary as function composition, and it was not clear for a long time how to capture the structure of probability theory categorically.

The resolution comes through the notion of a *Markov category*, introduced by Fritz (2020) building on earlier work of Giry and Golubtsov. The key insight is to work not with probability measures themselves but with *Markov kernels* as the morphisms of a category, and to identify the categorical properties that characterise the probabilistic structure.

Definition 6.1 (Markov Category). A *Markov category* is a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ in which every object A is equipped with a *copy morphism* $\delta_A : A \rightarrow A \otimes A$ and a *delete morphism* $\varepsilon_A : A \rightarrow I$, such that these maps are natural in A and satisfy the conditions of a *commutative comonoid*: $(\text{id}_A \otimes \delta_A) \circ \delta_A = (\delta_A \otimes \text{id}_A) \circ \delta_A$ (coassociativity), $(\varepsilon_A \otimes \text{id}_A) \circ \delta_A = \text{id}_A$ (left unit), and the symmetry condition $\sigma_{A,A} \circ \delta_A = \delta_A$ (cocommutativity).

Moreover, for every morphism $f : A \rightarrow B$ in $\mathcal{C}$, the morphism f is required to preserve deletion: $\varepsilon_B \circ f = \varepsilon_A$, expressing the normalisation of probability: a stochastic kernel applied to any state must produce a probability measure (not just a sub-probability measure).

The canonical example is **BorelStoch**: the category whose objects are standard Borel spaces (Polish topological spaces with their Borel σ -algebra) and whose morphisms $f : X \rightarrow Y$ are Markov kernels — measurable functions $f(-|x) : Y \rightarrow [0, 1]$ that form a probability measure on Y for each $x \in X$. Composition of kernels is convolution (integration), and the monoidal product on objects is the cartesian product of measurable spaces. The copy

morphism $\delta_X : X \rightarrow X \times X$ is the kernel $\delta_X(A \times B|x) = \mathbf{1}_{x \in A} \mathbf{1}_{x \in B}$ (the Dirac delta kernel that copies x into both components), and the delete morphism $\varepsilon_X : X \rightarrow 1$ sends everything to the unique point.

Note the contrast with cartesian categories: in a cartesian category, every morphism is a (deterministic) function and the copy map is the diagonal. In a Markov category, the copy map expresses the ability to observe a random variable twice and get the same value, while stochastic morphisms need not preserve copying. The condition that morphisms preserve deletion (normalisation) is precisely what distinguishes proper probability kernels from sub-probability kernels (which would give rise to a weaker structure called a *CD-category*).

6.2 Conditional Independence and Bayesian Networks

One of the most satisfying applications of Markov categories is the categorical treatment of conditional independence. In classical probability theory, $X \perp\!\!\!\perp Y \mid Z$ (“ X is conditionally independent of Y given Z ”) is defined analytically: $p(X, Y|Z) = p(X|Z)p(Y|Z)$. In a Markov category, this can be expressed purely in terms of the categorical structure, without reference to measures or densities.

Specifically, in a Markov category $\mathcal{C}$, a morphism $f : Z \rightarrow X \otimes Y$ expresses a joint distribution over (X, Y) given Z . Conditional independence $X \perp\!\!\!\perp Y \mid Z$ holds if and only if f factors as $(f_X \otimes f_Y) \circ \delta_Z$, where $f_X : Z \rightarrow X$ and $f_Y : Z \rightarrow Y$ are the marginals and $\delta_Z : Z \rightarrow Z \otimes Z$ is the copy map. In string diagram notation: the joint distribution is the same as the product of the two marginals, both of which depend on (a copy of) Z .

This categorical characterisation subsumes the classical analytic definition and applies uniformly to discrete, continuous, and mixed distributions (and even to non-standard probability theories that fit the Markov category framework but do not correspond to classical measure-theoretic probability). It also applies without modification to quantum probability, where the copy map is replaced by the quantum copying operation.

Theorem 6.2 (Fritz, 2020). *A symmetric monoidal string diagram equation between morphisms in **BorelStoch** holds for all standard Borel spaces if and only if it is derivable from the axioms of Markov categories.*

This is the Markov category analogue of Mac Lane’s coherence theorem: one can reason about probabilistic processes using string diagrams, and any string diagram equation that is derivable from the axioms holds universally.

6.3 Bayesian Inference as a Morphism

Bayesian inference — the computation of a posterior distribution from a prior and a likelihood — can be expressed as a morphism in a Markov category, and the conditions under which Bayesian inference is possible (the existence of a disintegration) have a precise categorical characterisation.

Given a joint morphism $f : I \rightarrow X \otimes Y$ (a joint distribution over X and Y , with I the monoidal unit), the *marginal* on X is $f_X = (\text{id}_X \otimes \varepsilon_Y) \circ f : I \rightarrow X$, and the *conditional*

$f_{Y|X} : X \rightarrow Y$ (if it exists) satisfies $f = (f_{Y|X} \otimes \text{id}_X) \circ \delta_X \circ f_X$ (the joint is the product of the prior on X with the conditional kernel from X to Y). The existence of such a conditional — a *disintegration* of the joint distribution — is guaranteed in **BorelStoch** for all standard Borel spaces (a non-trivial measure-theoretic result, the disintegration theorem), and its categorical formulation makes its universality transparent.

Bayesian updating is then the operation of replacing the prior f_X with the posterior $f_{X|Y}$, given an observation $y \in Y$. Categorically, this is the image of the morphism $f : I \rightarrow X \otimes Y$ under a certain natural transformation that swaps the roles of X and Y in the disintegration. The resulting morphism $f_{X|Y} : Y \rightarrow X$ maps each observed value y to the posterior distribution over X . The Bayes-optimal prediction rule is then the composite $\hat{X}(y) = \text{argmax}_x f_{X|Y}(x|y)$, which is a deterministic morphism $Y \rightarrow X$ obtained by applying the MAP operator to the posterior.

6.4 Variational Inference and the Evidence Lower Bound

Variational autoencoders (VAEs) and other variational inference methods can be understood categorically as attempts to approximate an intractable posterior $f_{X|Y}$ by a tractable “variational” posterior $q_{X|Y}$ drawn from a parametric family. The Evidence Lower Bound (ELBO), which is the objective optimised in variational inference, has a natural categorical expression.

In a Markov category equipped with a measurable structure (specifically, in **BorelStoch**), the *KL divergence* from q to p (both morphisms $I \rightarrow X$ in $\mathcal{C}$) can be defined intrinsically as:

$$\text{KL}(q||p) = \mathbb{E}_{x \sim q} \left[\log \frac{dq}{dp}(x) \right],$$

provided the Radon-Nikodym derivative exists (i.e., $q \ll p$). This quantity, while analytic, arises naturally in the string diagram calculus as a measure of the “information distance” between two paths from I to X .

The ELBO can then be written as:

$$\mathcal{L}(q) = \mathbb{E}_{x \sim q_{X|Y}(y)} [\log p_{Y|X}(y|x)] - \text{KL}(q_{X|Y}(y)||p_X),$$

where the first term rewards the variational posterior for placing mass on values of X that explain the observation $Y = y$ under the generative model $p_{Y|X}$, and the second term penalises the variational posterior for deviating from the prior p_X . The ELBO is a lower bound on the log-evidence $\log p(y)$ (by Jensen’s inequality applied to the concave log function), and maximising the ELBO is equivalent to minimising $\text{KL}(q||p_{X|Y})$ when the generative model p is fixed.

In the Markov category framework, the ELBO admits a clean categorical interpretation: it is the morphism-level analogue of a certain adjunction between the “generative” direction (from latent to observed) and the “inference” direction (from observed to latent). The VAE training procedure is an attempt to make this adjunction as tight as possible within the constraints of the parametric families chosen for both directions. The category-theoretic

language makes it clear that the ELBO is not an arbitrary objective but the natural one associated with the Bayesian adjunction.

Chapter 7

Enriched Categories, Metric Spaces, and Robustness

7.1 Enrichment and the Quantification of Structure

Ordinary category theory talks about *sets* of morphisms: between any two objects A and B there is a *set* $\mathcal{C}(A, B)$. But in many contexts of interest, the morphisms carry additional structure: they form a topological space (as in topologically enriched categories), an abelian group (as in **Ab**-enriched or *preadditive* categories), a vector space (as in **Vect**-enriched or *linear* categories), a metric space (as in $[0, \infty]$ -enriched categories), or a category (as in bicategories or 2-categories). The notion of an *enriched category* captures this additional structure uniformly.

Definition 7.1 (Enriched Category). Let $(\mathcal{V}, \otimes, I)$ be a symmetric monoidal category. A $\mathcal{V}$ -enriched category $\mathcal{C}$ consists of:

1. a collection $\text{ob}(\mathcal{C})$ of objects,
2. for each pair (A, B) of objects, an object $\mathcal{C}(A, B) \in \text{ob}(\mathcal{V})$ (the *hom-object*),
3. for each triple (A, B, C) , a morphism in $\mathcal{V}$:

$$\circ_{A,B,C} : \mathcal{C}(B, C) \otimes \mathcal{C}(A, B) \rightarrow \mathcal{C}(A, C) \quad (\text{composition}),$$

4. for each object A , a morphism $j_A : I \rightarrow \mathcal{C}(A, A)$ in $\mathcal{V}$ (the *identity*),

satisfying associativity and unit coherence diagrams in $\mathcal{V}$.

When $\mathcal{V} = \mathbf{Set}$ with cartesian product, a **Set**-enriched category is just an ordinary category. When $\mathcal{V} = \mathbf{Ab}$ (abelian groups), we get a preadditive category, whose morphism sets are abelian groups and whose composition is bilinear. When $\mathcal{V} = \mathbf{FVect}_{\mathbb{R}}$, we get a linear category.

The enrichment most directly relevant to robustness and metric structure is due to Lawvere's remarkable observation (1973) that *metric spaces are precisely categories enriched*

in $([0, \infty], \geq, +, 0)$, where $[0, \infty]$ is the poset of extended non-negative reals (ordered by $\geq$, so that 0 is the top and ∞ is the bottom) with the monoidal structure given by addition.

7.2 Lawvere Metric Spaces

Theorem 7.2 (Lawvere, 1973). *A (generalised) metric space (in the sense of a function $d : X \times X \rightarrow [0, \infty]$ satisfying $d(x, x) = 0$ and the triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$, but not necessarily symmetry or the condition $d(x, y) = 0 \Rightarrow x = y$) is the same as a category enriched in $([0, \infty], \geq, +, 0)$: the objects are the points of X , the hom-objects are $d(x, y) \in [0, \infty]$ (with the convention that there is a morphism from x to y if and only if $d(x, y) < \infty$), composition is the triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$, and the identity at x is the condition $d(x, x) = 0$.*

A $[0, \infty]$ -enriched functor between Lawvere metric spaces is a *non-expansive map*: a function $f : X \rightarrow Y$ with $d_Y(f(x), f(x')) \leq d_X(x, x')$. A $[0, \infty]$ -enriched natural transformation between two such functors $f, g : X \rightarrow Y$ is a “uniform shift”: a scalar $\varepsilon \geq 0$ such that $d_Y(f(x), g(x)) \leq \varepsilon$ for all x , which is one way to formalise the closeness of two functions.

The immediate application to deep learning is through the notion of *Lipschitz stability*. A neural network layer $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is Lipschitz continuous with constant L if $\|f(x) - f(x')\| \leq L\|x - x'\|$ for all x, x' . This is precisely the condition that f is a non-expansive map between the Lawvere metric spaces $(\mathbb{R}^m, L^{-1}\|\cdot\|)$ and $(\mathbb{R}^n, \|\cdot\|)$ (or, equivalently, that f is a Lipschitz- L map between the standard metric spaces). The composition of Lipschitz maps is Lipschitz with the product of the constants, corresponding to the enriched composition: $d(f \circ g) \leq d(f) \cdot d(g)$ in the enriched sense (with multiplication replacing addition in the appropriate enrichment).

7.3 Enriched Functors as Robustness Certificates

The categorical framework suggests the following robustness programme: a neural network architecture that is formulated as an enriched functor between Lawvere metric spaces (or, more generally, between categories enriched over an appropriate quantale) comes with an automatic robustness guarantee, because enriched functors preserve the metric structure.

Concretely: if a network $F : \mathcal{C} \rightarrow \mathcal{D}$ is an enriched functor between two $[0, \infty]$ -enriched categories (Lawvere metric spaces), then for any two inputs x, x' :

$$d_{\mathcal{D}}(F(x), F(x')) \leq d_{\mathcal{C}}(x, x').$$

This is the Lipschitz-1 condition. The architecture design problem becomes: how can we parametrise a family of enriched functors? Spectral normalisation (constraining the spectral norm of weight matrices to be at most 1) is one answer for linear maps; the full theory of enriched functors in the Lawvere metric setting gives a general framework that extends to nonlinear operations, normalisation layers, and attention.

The connection to adversarial robustness is immediate: a network that is an enriched functor with Lipschitz constant L cannot be fooled by adversarial perturbations of size ε

more than $L\varepsilon$ in output space, because the enriched functor condition bounds the change in output by L times the change in input.

7.4 Quantale-Enriched Categories and Generalised Metrics

A *quantale* is a complete lattice Q with an associative binary operation $*$ that distributes over arbitrary joins: $a * (\bigvee_i b_i) = \bigvee_i (a * b_i)$. The archetypal examples are:

- $([0, \infty], \geq, +)$: the Lawvere quantale for metric spaces,
- $([0, 1], \geq, \cdot)$: the multiplicative quantale, giving non-expansive maps for the “multiplicative” metric,
- $(\{0, 1\}, \geq, \wedge)$: the Boolean quantale, giving ordinary categories (enrichment over the Boolean quantale recovers ordinary posets).

Categories enriched over different quantales give rise to different notions of “distance” or “closeness” between morphisms, and the choice of quantale corresponds to the choice of error metric for robustness. Enrichment over $([0, \infty], \geq, +)$ gives Lipschitz robustness (additive error accumulates additively through composition); enrichment over $([0, 1], \geq, \cdot)$ gives robustness under multiplicative error (error is bounded multiplicatively); enrichment over a probabilistic quantale gives stochastic robustness guarantees.

The categorical framework thus provides a unified language for different notions of robustness, making precise the conditions under which robustness is preserved under composition (functoriality of the enriched functor) and under the operations of the architecture (the monoidal structure of the enriched category).

Chapter 8

The Yoneda Lemma and the Geometry of Representation

8.1 Representable Functors

The Yoneda lemma is the deepest result in elementary category theory. It tells us that every object in a category is completely determined by its relationships with all other objects, and that the set of natural transformations from a representable functor is in natural bijection with the values of any functor at the representing object. This is a powerful abstraction, and its manifestations in deep learning — from embedding methods to the neural tangent kernel to the theory of attention — are numerous and non-trivial.

For any category $\mathcal{C}$ and any object $A \in \text{ob}(\mathcal{C})$, the *representable functor* $\mathcal{C}(A, -) : \mathcal{C} \rightarrow \mathbf{Set}$ sends each object B to the set $\mathcal{C}(A, B)$ of morphisms from A to B , and sends each morphism $f : B \rightarrow C$ to the function $f \circ - : \mathcal{C}(A, B) \rightarrow \mathcal{C}(A, C)$, $g \mapsto f \circ g$. This is the covariant Yoneda embedding. Similarly, the *contravariant Yoneda embedding* is $\mathcal{C}(-, A) : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}$.

Theorem 8.1 (Yoneda Lemma). *For any functor $F : \mathcal{C} \rightarrow \mathbf{Set}$ and any object $A \in \text{ob}(\mathcal{C})$, there is a natural bijection:*

$$\text{Nat}(\mathcal{C}(A, -), F) \cong F(A),$$

where Nat denotes the set of natural transformations. This bijection is natural in both A (contravariantly) and F (covariantly).

Proof. Given a natural transformation $\eta : \mathcal{C}(A, -) \Rightarrow F$, its component at A is a function $\eta_A : \mathcal{C}(A, A) \rightarrow F(A)$, and we assign to η the element $\eta_A(\text{id}_A) \in F(A)$.

Conversely, given $x \in F(A)$, we define the natural transformation $\eta^x : \mathcal{C}(A, -) \Rightarrow F$ by $\eta_B^x(f) = F(f)(x)$ for each $f \in \mathcal{C}(A, B)$. Naturality of η^x follows from the functoriality of F : for $g : B \rightarrow C$, $\eta_C^x(g \circ f) = F(g \circ f)(x) = F(g)(F(f)(x)) = F(g)(\eta_B^x(f))$.

The two constructions are mutually inverse: $\eta_B^{\eta_A(\text{id}_A)}(f) = F(f)(\eta_A(\text{id}_A)) = \eta_B(f)$ by naturality of η applied to $f : A \rightarrow B$ and id_A , and $\eta_A^x(\text{id}_A) = F(\text{id}_A)(x) = x$.

Naturality of the bijection in A and F is verified similarly. $\square$

The Yoneda lemma has an immediate corollary: the functor $A \mapsto \mathcal{C}(A, -)$ (the Yoneda embedding $\mathcal{Y} : \mathcal{C}^{\text{op}} \rightarrow [\mathcal{C}, \mathbf{Set}]$) is fully faithful, meaning that $\mathcal{C}(A, B) \cong \text{Nat}(\mathcal{C}(B, -), \mathcal{C}(A, -))$. Two objects are isomorphic in $\mathcal{C}$ if and only if the functors they represent are naturally isomorphic. Thus, an object is determined, up to isomorphism, by what morphisms go out of it.

8.2 The Yoneda Perspective on Embeddings

Word embeddings, graph embeddings, and other representation learning methods in deep learning can be understood through the Yoneda lens. The fundamental idea of an embedding is to represent each object (a word, a node, an image) as a vector in $\mathbb{R}^d$, in such a way that the geometric relationships between vectors reflect the structural relationships between objects. This is precisely what the Yoneda embedding does: it represents each object A by the functor $\mathcal{C}(A, -)$, which encodes all the relationships of A to every other object.

The word2vec model can be understood as follows. Consider the category whose objects are words in a vocabulary and whose morphisms $w \rightarrow w'$ are the co-occurrence statistics (the probability that w' appears near w in a text corpus). This is a category enriched over $[0, 1]$ (since co-occurrence probabilities lie in $[0, 1]$). The Yoneda embedding then assigns to each word w the vector of its co-occurrence probabilities with all other words in the vocabulary — the row of the co-occurrence matrix corresponding to w . Word2vec training finds a low-dimensional factorisation of this co-occurrence matrix, approximating the Yoneda representation by a vector of dimension $d \ll |V|$.

The *noise contrastive estimation* (NCE) objective used in training word2vec can be interpreted categorically as minimising the distance between the Yoneda representation and its low-dimensional approximation, measured in a suitable enriched sense.

8.3 Presheaves and Universal Representations

The Yoneda embedding $\mathcal{Y} : \mathcal{C} \rightarrow [\mathcal{C}^{\text{op}}, \mathbf{Set}]$ lands in the *presheaf category* $[\mathcal{C}^{\text{op}}, \mathbf{Set}]$, whose objects are functors $F : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}$ (presheaves) and whose morphisms are natural transformations. A presheaf on $\mathcal{C}$ is a set-valued contravariant functor on $\mathcal{C}$: for each object A , it assigns a set $F(A)$, and for each morphism $f : A \rightarrow B$, a function $F(f) : F(B) \rightarrow F(A)$ (going backwards).

The presheaf category $[\mathcal{C}^{\text{op}}, \mathbf{Set}]$ is the *free cocompletion* of $\mathcal{C}$: it is the universal category with all small colimits in which $\mathcal{C}$ embeds. Every small colimit in $\mathbf{Set}$ (and more generally in any cocomplete category $\mathcal{D}$) can be computed in $[\mathcal{C}^{\text{op}}, \mathbf{Set}]$ and then transported along a functor. The Yoneda embedding preserves limits (it is right adjoint to the “evaluation at generators” functor) but not colimits: the colimit of a diagram in $[\mathcal{C}^{\text{op}}, \mathbf{Set}]$ is computed pointwise, but the Yoneda embedding of the colimit in $\mathcal{C}$ is not the pointwise colimit of the Yoneda embeddings.

This has a direct interpretation for deep learning: the presheaf category is the “ideal” repre-

sensation space in which every structural relationship can be faithfully represented, without any loss of information. Practical embedding methods choose a finite-dimensional approximation $\mathbb{R}^d$ to this ideal representation, and the quality of the approximation depends on how much of the presheaf structure can be captured in d dimensions. The dimension d plays the role of the “expressivity” parameter of the representation, and the gap between the ideal presheaf representation and the low-dimensional one is the information-theoretic content of the representation learning problem.

8.4 The Neural Tangent Kernel as a Representable Functor

The *neural tangent kernel* (NTK) of a neural network, introduced by Jacot, Gabriel, and Hongler (2018), is the kernel function:

$$K_\theta(x, x') = \left\langle \frac{\partial f_\theta(x)}{\partial \theta}, \frac{\partial f_\theta(x')}{\partial \theta} \right\rangle,$$

where $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$ is the network output, θ is the parameter vector, and the inner product is taken over the parameter space. The NTK describes the dynamics of gradient descent in the infinite-width limit: under suitable overparametrisation, the NTK converges to a fixed kernel K at initialisation and remains constant throughout training.

The Yoneda perspective on the NTK is the following. Consider the category $\mathcal{C}_\theta$ whose objects are input points $x \in \mathbb{R}^d$ and whose hom-objects are the gradients $\mathcal{C}_\theta(x, x') = \nabla_\theta f_\theta(x)$ (as vectors in parameter space). The NTK is then:

$$K_\theta(x, x') = \langle \mathcal{C}_\theta(x, \cdot), \mathcal{C}_\theta(x', \cdot) \rangle_{L^2}$$

— the inner product of the “representable functors” (gradient vectors) at x and x' . The NTK gram matrix $K_\theta(x_i, x_j)$ is the Gram matrix of the Yoneda embedding of the training data points into the parameter space (via gradients).

This perspective immediately explains why the NTK governs generalisation: the Yoneda embedding is the most informative representation of a point, and the NTK measures how similar two points are in this maximal representation. The ridge regression with the NTK as kernel is the Bayesian posterior under a Gaussian prior on functions, and this is the function that gradient descent converges to in the infinite-width limit.

8.5 Attention as a Categorical Inner Product

The self-attention mechanism of the Transformer can be understood as a categorification of the inner product structure that appears in the Yoneda embedding and the NTK. Recall that self-attention computes:

$$\text{Attn}(X) = \text{softmax}\left(\frac{XW_Q(XW_K)^\top}{\sqrt{d_k}}\right)XW_V,$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$ are learned projection matrices. The inner product $XW_Q(XW_K)^\top$ computes, for each pair of positions (i, j) , the inner product of the query at position i with the key at position j : this is exactly a (learned, projected) inner product between the Yoneda representations of positions i and j in the sequence.

The softmax normalises these inner products into weights, and the output is a weighted combination of the value vectors. The attention mechanism thus computes, for each query position i , a weighted average of the Yoneda representations of all other positions (weighted by their inner product with position i), projected into value space. This is a practical approximation to the ideal Yoneda representation of the structural relationships between positions in the sequence.

The categorical content of this observation is the following: the attention mechanism is computing the *coend* $\int^j K(i, j) \cdot V_j$, where $K(i, j) = \text{softmax}(\langle q_i, k_j \rangle)$ is the attention weight (a measure of how much position j 's representation should contribute to position i 's output) and $V_j = xW_V$ is the value at position j . The coend computes a kind of “weighted colimit”, integrating over all positions with weights given by the attention kernel. In the limit of infinite sequence length, this integral converges (under appropriate assumptions) to an integral transform with kernel K , and the attention mechanism is an approximation to this ideal integral transform computed on the finite sample of positions in the sequence.

Chapter 9

Topoi, Sheaves, and the Logic of Generalisation

9.1 Grothendieck Topoi

A *Grothendieck topos* is a category $\mathcal{E}$ that is equivalent to the category of sheaves on some site. While this definition requires unpacking, the important point for our purposes is that topoi provide a setting in which the full power of set-theoretic reasoning — including the internal logic of the topos — can be developed in a way that is parameterised by the geometric or combinatorial structure of the underlying site.

Definition 9.1 (Grothendieck Topos). A *site* is a category $\mathcal{C}$ equipped with a *Grothendieck topology* J : an assignment of “covering families” $J(A) = \{(f_i : A_i \rightarrow A)_{i \in I}\}$ to each object A , satisfying stability, local character, and identity axioms. A *sheaf* on the site $(\mathcal{C}, J)$ is a presheaf $F : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}$ satisfying: for every covering family $\{f_i : A_i \rightarrow A\}$ in $J(A)$, the diagram

$$F(A) \rightarrow \prod_i F(A_i) \rightrightarrows \prod_{i,j} F(A_i \times_A A_j)$$

is an equaliser. The category $\mathbf{Sh}(\mathcal{C}, J)$ of sheaves on $(\mathcal{C}, J)$ is a Grothendieck topos.

The sheaf condition expresses a locality principle: the value of a sheaf on an object A is completely determined by its values on a covering of A , and any compatible family of local sections glues uniquely to a global section. This is the categorical expression of the principle that global properties are determined by consistent local information.

The most basic example is the topos of sheaves on a topological space X (with the usual open cover topology). A sheaf on X is a continuous “bundle of sets” over X , and the sheaf condition says that a global section is the same as a compatible system of local sections on an open cover. The category of sheaves on X is a Grothendieck topos.

For machine learning, the relevant sites are typically: the site of finite sets (giving the topos of presheaves on **FinSet**, related to combinatorial models of data), the site of smooth manifolds (giving the topos of sheaves on **Man**, related to differential-geometric models), and the site of probability spaces (giving a topos related to measure-theoretic models).

Each of these topoi comes with its own internal logic, and reasoning within that logic corresponds to working within the chosen model of data.

9.2 The Internal Language of a Topos

Every Grothendieck topos has an *internal language*, a dependent type theory in which one can reason about the objects and morphisms of the topos as if they were sets and functions, but with the semantics given by the topos structure. This internal language is called the *Mitchell-Bénabou language* or the *internal type theory* of the topos.

The key feature is that the topos has a *subobject classifier* Ω : an object that plays the role of the set $\{0, 1\}$ (truth values) in **Set**, but whose elements are the possible “truth values” in the topos. In **Set**, the subobject classifier is $\{0, 1\}$ and the classifying map $\chi_S : A \rightarrow \{0, 1\}$ of a subset $S \subseteq A$ is the indicator function. In the topos of sheaves on a topological space X , the subobject classifier is the sheaf of open subsets of X : a truth value at a point $x \in X$ is an open neighbourhood of x (representing how “locally true” a statement is at x).

The connection to generalisation theory is subtle but real. In learning theory, a hypothesis $h : X \rightarrow \{0, 1\}$ is a deterministic classifier, and the *generalisation gap* measures the difference between training accuracy and test accuracy. In the topos-theoretic framework, we can replace the Boolean truth values $\{0, 1\}$ with the richer truth values of a topos, and “truth” becomes a local, contextual notion. A classifier that is “true” on the training data (a global section of a sheaf on the data site) can be “locally true” in a neighbourhood of the training data but fail to extend globally — this is overfitting, captured by the failure of the sheaf condition.

9.3 Sheaves and Compositional Generalisation

Compositional generalisation — the ability of a model to correctly predict the behaviour of novel combinations of known components — is one of the central challenges of deep learning. Systems that learn independently about objects and their properties should, in principle, be able to generalise to novel combinations. In practice, most deep networks fail at this: they generalise within the training distribution but not to systematic compositional variations.

The sheaf-theoretic framework provides a precise formulation of compositional generalisation. Consider a dataset that consists of descriptions of objects (e.g., sentences describing combinations of objects and attributes). A compositionally generalising model should be described by a sheaf F on the “compositionality site” — the site whose objects are the possible descriptions and whose covers represent the ways in which descriptions can be built from components. A section of the sheaf F assigns a representation to each description, and the sheaf condition requires that the representation of a composite description is determined (“glues”) from the representations of its components.

A model that satisfies the sheaf condition would automatically compositionally generalise: given representations for the components, the representation of their composite is deter-

mined by the covering relation. Models that fail at compositional generalisation — that is, most current neural networks — fail to satisfy the sheaf condition: their representations of composites are not determined by their representations of components in any consistent way.

The sheaf perspective suggests a structural remedy: design architectures that are forced, by their structure, to produce sheaf-consistent representations. The compositional structure of a natural language (the parse tree of a sentence, for instance) defines a natural sheaf structure on linguistic expressions, and a model that respects this sheaf structure would automatically compositionally generalise within the class of expressions admitted by the grammar. This is the categorical generalisation of the principle behind recursive neural networks and tree-structured models.

9.4 Geometric Morphisms and Transfer Learning

A *geometric morphism* between topoi $f : \mathcal{E} \rightarrow \mathcal{F}$ is a pair of functors (f^*, f_*) with $f^* \dashv f_*$ (an adjunction), where $f^* : \mathcal{F} \rightarrow \mathcal{E}$ (inverse image) is left exact (preserves finite limits). Geometric morphisms are the morphisms of the 2-category of Grothendieck topoi, and they encode “continuous” maps between the geometric objects represented by the topoi.

In the learning context, a geometric morphism from $\mathcal{E}_{\text{source}}$ to $\mathcal{E}_{\text{target}}$ (where each topos represents a “world” of possible data distributions) provides a structured way to transfer representations from the source to the target. The inverse image functor f^* pulls back structures from the target topos to the source, while the direct image functor f_* pushes forward. Transfer learning is, in this framework, the problem of finding a geometric morphism from the learned source topos to the target topos, or equivalently, of finding a structural map that is compatible with the sheaf structures on both sides.

The key constraint that geometric morphisms preserve finite limits corresponds, in the learning setting, to the requirement that the transfer map preserves the logical structure of descriptions: a model trained on a source task that compositionally relates attributes to objects should, via the geometric morphism, produce a target model that relates the corresponding target attributes to target objects. Fine-tuning, prompting, and other transfer learning techniques can be analysed in terms of how well they approximate a genuine geometric morphism.

Chapter 10

Categorical Cybernetics and Open Games

10.1 Systems with Agency

The frameworks developed in previous chapters treat neural networks as morphisms — passive transformers of data. But many learning scenarios involve *agency*: a system that chooses actions based on observed context, seeking to optimise an objective. Reinforcement learning, game theory, and multi-agent learning all involve such agentic systems. The categorical theory of these scenarios is provided by *categorical cybernetics* and the theory of *open games*.

Definition 10.1 (Open Game). An *open game* $G = (X, S, \Sigma, P, E)$ consists of:

1. a *context* type X (the observations available to the players),
2. a *outcome* type S (the signals received from the environment after the game),
3. a *strategy type* Σ (the possible strategies for the player),
4. a *play function* $P : \Sigma \times X \rightarrow A$ (how a strategy determines an action given a context),
5. an *equilibrium function* $E : X \times S^A \rightarrow \mathcal{P}(\Sigma)$ (which strategies are best responses given a context and a function from actions to downstream outcomes).

Open games compose: given $G_1 : (X_1, S_1) \rightarrow (A_1, R_1)$ and $G_2 : (X_2, S_2) \rightarrow (A_2, R_2)$, their sequential composite $G_2 \circ G_1$ represents a situation where the first player acts, the second player observes the first player’s action, and the equilibrium of the composite is determined by the joint equilibria of both players.

Theorem 10.2 (Hedges, 2018; Ghani–Hedges–Winschel–Zahn, 2018). *Open games, with their composition operation, form a symmetric monoidal category **Game**. Sequential composition corresponds to player 2 observing player 1’s action; parallel composition corresponds to two independent players acting simultaneously.*

The relationship to deep learning arises through *gradient play*: a neural network trained by gradient descent can be viewed as an open game where the “player” is the optimiser and

the “strategy” is the choice of network parameters. The “equilibrium” of this game is the set of local minima of the loss function, and the gradient descent algorithm is a particular strategy-update rule for this player. From this perspective, the dynamics of stochastic gradient descent in a deep network are governed by the equilibrium theory of an open game, and the properties of the optima — their generalisation behaviour, their sharpness, their stability to perturbation — are properties of the equilibria of this game.

10.2 Categorical Cybernetics

Capucci and Gavranović (2022) developed a framework called *categorical cybernetics* that unifies lenses, optics, games, and learning algorithms under a single categorical umbrella. The central notion is that of a *bidirectional system*: a system that has both a forward direction (computing outputs from inputs) and a backward direction (computing “corrections” or “feedback” from outputs to inputs).

The key observation is that every bidirectional system can be described as a morphism in a suitable *actegory* — a category $\mathcal{C}$ together with an action of a monoidal category $\mathcal{M}$ on $\mathcal{C}$. The lenses of Chapter 3 arise as morphisms in the actegory of the cartesian product action; the optics of Chapter 4 arise as morphisms in the actegory of the tensor product action; open games arise as morphisms in the actegory of the game monad action.

The unifying principle is that the bidirectionality of a system corresponds to the structure of the action: the forward direction is the action of $\mathcal{M}$ on $\mathcal{C}$, and the backward direction is the co-action (the adjoint of the action in the appropriate sense). Different choices of action give different notions of bidirectionality, all instances of the same categorical pattern.

10.3 Gradient Descent as a Morphism in Categorical Cybernetics

In the categorical cybernetics framework, gradient descent training of a neural network is a morphism from the “parameter” object P to itself, in a category that captures the iterative structure of the training loop. Specifically:

Let $\mathcal{C}$ be the category **Smooth** with its cartesian monoidal structure. A parametrised neural network is a morphism $(P, f) : A \rightarrow B$ in **Para**($\mathcal{C}$). The loss function $\ell : B \times \hat{B} \rightarrow \mathbb{R}$ (where $\hat{B}$ is the label space) gives a morphism in $\mathcal{C}$. The gradient of the loss with respect to the parameters is the composition of the optic of f (Chapter 4) with the gradient of ℓ : it is a morphism $\nabla_P \ell : P \times A \times \hat{B} \rightarrow T^*P$ (from parameter-input-label triples to the cotangent of parameter space).

A single gradient descent step is then the morphism $\text{SGD} : P \times A \times \hat{B} \rightarrow P$ defined by $\text{SGD}(p, a, b) = p - \eta \nabla_P \ell(p, a, b)$, where $\eta > 0$ is the learning rate. The training loop iterates this morphism over the dataset: it is the morphism obtained by applying SGD repeatedly, i.e., the n -fold iterate $\text{SGD}^n : P \times (A \times \hat{B})^n \rightarrow P$.

The categorical cybernetics perspective reveals that the training loop is not merely a sequence of gradient steps but a morphism in a *traced monoidal category*: the “trace” of the

training loop is the feedback from the updated parameters back into the next gradient computation. The existence of this trace corresponds, in the physics literature, to the existence of a fixed point of the gradient dynamics (a stationary point of the loss), and the trace axioms correspond to the conditions under which such a fixed point exists.

Chapter 11

Free Probability, the Composition of Random Layers, and the Freeness Conjecture

11.1 Classical vs. Free Independence

Classical probability theory studies random variables on a common probability space, and the notion of independence: X and Y are independent if $\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\mathbb{E}[g(Y)]$ for all bounded measurable f, g . Free probability theory, introduced by Voiculescu in the 1980s, studies random variables in a non-commutative probability space — a pair (A, τ) where A is a unital C^* -algebra and $\tau : A \rightarrow \mathbb{C}$ is a faithful trace state. In this non-commutative setting, the analogue of independence is *free independence* (or *freeness*).

Two self-adjoint elements $a, b \in A$ are *free* if, for any polynomials $p_1, q_1, p_2, q_2, \dots$ with $\tau(p_i(a)) = 0$ and $\tau(q_j(b)) = 0$:

$$\tau(p_1(a)q_1(b)p_2(a)q_2(b)\cdots) = 0.$$

Freeness is the correct analogue of independence for non-commutative random variables, and it has a rich combinatorial theory — the free cumulants — that exactly parallels the classical cumulant theory.

The relevance to deep learning is the following: the weight matrices of a randomly initialised neural network are random matrices (elements of the matrix algebra $M_n(\mathbb{R})$ with the normalised trace $\tau = \frac{1}{n}\text{tr}$), and the empirical spectral distribution (ESD) of the product $W_L W_{L-1} \cdots W_1$ (the composition of all weight matrices) is determined, in the large-width limit, by the free convolution of the individual ESDs. This is the content of the *freeness conjecture* for deep networks, which has been verified empirically and partially proved in special cases.

Theorem 11.1 (Freeness Conjecture, Pennington–Wakhloo, 2017; partial rigorous results in Benigni–Péché–Schnelli). *Let $W_1, \dots, W_L \in M_n(\mathbb{R})$ be independent random matrices from rotationally invariant distributions (e.g., i.i.d. Gaussian entries). Then, as $n \rightarrow \infty$:*

1. $W_1, \dots, W_L$ are asymptotically free in $(M_n(\mathbb{R}), \frac{1}{n}\text{tr})$,
2. the empirical spectral distribution of $\frac{1}{n}W_L \cdots W_1(W_L \cdots W_1)^\top$ converges to the free multiplicative convolution $\mu_1 \boxtimes \cdots \boxtimes \mu_L$ of the individual spectral distributions μ_i .

The categorical content of this theorem lies in the relationship between free probability and category theory. The free product of non-commutative probability spaces is a categorical colimit in the category of non-commutative probability spaces and trace-preserving unital maps. Freeness is the universal property of the free product: two subalgebras A_1 and A_2 of a non-commutative probability space (A, τ) are free if and only if the inclusion maps $A_1, A_2 \hookrightarrow A$ exhibit A as the free product of A_1 and A_2 (in the appropriate category).

This means that the freeness conjecture is, in categorical terms, the statement that the composition of independent random layers produces a non-commutative probability space that is the free product of the individual spaces. The Marchenko-Pastur law and the semicircle law of random matrix theory are the “free Gaussians” in this non-commutative probability space, and the free central limit theorem (freeness analogue of the classical CLT) is the statement that repeated free convolution converges to the semicircle distribution.

11.2 Categorical Consequences for Depth and Signal Propagation

The free probability treatment of deep networks has direct consequences for the signal propagation problem studied in the mean-field theory (cf. the companion volume on statistical physics). The correlation between two inputs x and x' at layer ℓ is determined, in the large-width limit, by the ℓ -fold free convolution of the weight spectrum, evaluated at the squared norm $\|x - x'\|^2$.

At the *edge of chaos* — the critical point between the ordered and chaotic phases of signal propagation — the spectral distribution of the weight matrices has specific shape properties (studied via the free entropy of Voiculescu). The categorical formulation makes clear that the edge of chaos is a *phase boundary* in the moduli space of non-commutative probability spaces equipped with trace-preserving maps, and that initialisation schemes designed to keep a network at the edge of chaos (e.g., He initialisation, LeCun initialisation) are choosing initial weights whose spectral distributions lie on this boundary.

Chapter 12

Categorical Limits, Colimits, and Universal Approximation

12.1 Limits and Colimits in Category Theory

A central theme of category theory is the study of *universal constructions*: objects defined by universal properties, meaning that every other object satisfying the relevant conditions factors uniquely through them. Limits and colimits are the two main families of universal constructions.

Given a functor $F : \mathcal{J} \rightarrow \mathcal{C}$ (a *diagram* of shape $\mathcal{J}$ in $\mathcal{C}$), the *limit* of F is an object $\lim F \in \mathcal{C}$ together with morphisms $\pi_j : \lim F \rightarrow F(j)$ for each $j \in \mathcal{J}$ (the *cone*), such that for any other cone (an object A with morphisms $\lambda_j : A \rightarrow F(j)$), there exists a unique morphism $u : A \rightarrow \lim F$ with $\pi_j \circ u = \lambda_j$ for all j . The colimit $\operatorname{colim} F$ is the dual: the initial cocone.

Special cases: the limit of the empty diagram is the terminal object (a singleton); the limit of a pair of objects A, B is their product $A \times B$; the limit of a parallel pair of morphisms $f, g : A \rightarrow B$ is their equaliser. The colimit of the empty diagram is the initial object; the colimit of a pair is the coproduct (disjoint union in **Set**); the colimit of a parallel pair is the coequaliser.

The universal approximation theorem for neural networks can be rephrased as a colimit statement: the class of functions computable by neural networks with a fixed architecture and variable parameters is *dense* (in an appropriate topology) in the space of continuous functions. In categorical terms: the functor from the category of parametrised networks to the category of continuous functions is “essentially surjective up to density”, meaning that every continuous function lies in the topological closure of the image of this functor. The precise formulation depends on the chosen topology (uniform, L^2 , etc.) and the domain of approximation.

12.2 Kan Extensions and the Universal Property of Learning

A *Kan extension* is perhaps the most universal construction in category theory. Given functors $F : \mathcal{C} \rightarrow \mathcal{E}$ and $K : \mathcal{C} \rightarrow \mathcal{D}$, the *left Kan extension* $\text{Lan}_K F : \mathcal{D} \rightarrow \mathcal{E}$ is the “best approximation” to F that factors through K :

$\text{Lan}_K F =$ the functor $G : \mathcal{D} \rightarrow \mathcal{E}$ with $G \circ K \cong F$, universal among all such.

The Kan extension exists, under appropriate conditions, as a colimit:

$$(\text{Lan}_K F)(d) = \int^{c \in \mathcal{C}} \mathcal{D}(K(c), d) \cdot F(c),$$

where $\cdot$ denotes the “tensoring” of an object $F(c) \in \mathcal{E}$ by a set $\mathcal{D}(K(c), d)$ (i.e., the $\mathcal{D}(K(c), d)$ -fold coproduct of $F(c)$ with itself, if $\mathcal{E}$ is cocomplete).

The relevance to learning: the problem of generalisation can be formulated as a Kan extension problem. Let $\mathcal{C}$ be the category of training data points and their relationships (e.g., similarity, label structure), $\mathcal{E}$ the category of output values (e.g., $\mathbb{R}^k$), $F : \mathcal{C} \rightarrow \mathcal{E}$ the “true” function we want to learn (defined only on training data), and $K : \mathcal{C} \rightarrow \mathcal{D}$ the inclusion of the training data into the full input space $\mathcal{D}$ (e.g., $\mathcal{D}$ is the category of all possible inputs). The Kan extension $\text{Lan}_K F : \mathcal{D} \rightarrow \mathcal{E}$ is then the universal extension of F from training data to all inputs — it is the function that is determined by the training data in the most general possible way, without any additional assumptions.

In the case where $\mathcal{D}$ has a topological or smooth structure, the Kan extension $\text{Lan}_K F$ is (under appropriate conditions) the interpolation that is consistent with F on training data and satisfies the weakest possible regularity conditions off the training data. This is related to the Reproducing Kernel Hilbert Space (RKHS) perspective on learning: the minimum-norm interpolant in an RKHS is a special case of a Kan extension, where the “shape” of the RKHS (determined by the kernel) corresponds to the specific enrichment of $\mathcal{D}$.

This connection suggests a categorical understanding of the *inductive bias* of a learning algorithm: it is the choice of the category $\mathcal{D}$ and its enrichment (or more generally, the choice of the category in which the Kan extension is computed). Different inductive biases correspond to different categories, and the quality of generalisation depends on how well the structure of $\mathcal{D}$ matches the true structure of the data-generating process.

Chapter 13

Monads, Comonads, and the Algebra of Effects

13.1 Monads from Adjunctions

Every adjunction $F \dashv G$ between categories $\mathcal{C}$ and $\mathcal{D}$ gives rise, canonically, to a *monad* on $\mathcal{C}$. The monad captures the “effect” of going from $\mathcal{C}$ to $\mathcal{D}$ and back: the composite $T = G \circ F : \mathcal{C} \rightarrow \mathcal{C}$ is an endofunctor equipped with a unit $\eta : \text{Id}_{\mathcal{C}} \Rightarrow T$ (from the unit of the adjunction) and a multiplication $\mu : T^2 \Rightarrow T$ (from the counit). The three axioms that (η, μ) must satisfy — associativity of μ and the two unit laws — follow automatically from the triangle identities of the adjunction.

Definition 13.1 (Monad). A *monad* on a category $\mathcal{C}$ is a triple (T, η, μ) where $T : \mathcal{C} \rightarrow \mathcal{C}$ is an endofunctor, $\eta : \text{Id}_{\mathcal{C}} \Rightarrow T$ is the *unit*, and $\mu : T \circ T \Rightarrow T$ is the *multiplication*, satisfying:

1. *Associativity*: $\mu \circ T\mu = \mu \circ \mu_T$ as natural transformations $T^3 \Rightarrow T$,
2. *Left unit*: $\mu \circ \eta_T = \text{id}_T$,
3. *Right unit*: $\mu \circ T\eta = \text{id}_T$.

In string diagram notation, a monad is a monoid object in the monoidal category $[\mathcal{C}, \mathcal{C}]$ of endofunctors of $\mathcal{C}$ with monoidal structure given by functor composition.

The string diagram perspective is illuminating: if we draw T as a wire and μ as a binary “merge” box (two wires entering, one leaving), then the associativity and unit axioms of the monad are exactly the axioms of a monoid written graphically — the same axioms, but now operating on a functor wire rather than a scalar. This is the precise sense in which monads are “monoids in the category of endofunctors”, a formulation due to Mac Lane.

The archetypal monad for deep learning is the *probability monad*. On **Set**, the probability monad $\mathcal{P}$ sends each set X to the set $\mathcal{P}(X)$ of probability distributions over X (say, finitely supported ones for concreteness). The unit $\eta_X : X \rightarrow \mathcal{P}(X)$ sends each element x to the Dirac measure δ_x . The multiplication $\mu_X : \mathcal{P}(\mathcal{P}(X)) \rightarrow \mathcal{P}(X)$ sends a distribution over distributions to its “flattened” or “integrated” version: a mixture $\sum_i p_i \nu_i$ is sent to

the marginal distribution $\sum_i p_i \nu_i \in \mathcal{P}(X)$. This flattening is precisely the law of total probability, and the monad axioms correspond to the consistency of iterated mixing.

Example 13.2 (Giry Monad). On the category **Meas** of measurable spaces and measurable maps, the *Giry monad* $\mathcal{G}$ sends each measurable space $(X, \mathcal{F})$ to the space $\mathcal{G}(X)$ of probability measures on $(X, \mathcal{F})$, equipped with the smallest σ -algebra making all evaluation maps $\nu \mapsto \nu(A)$ (for $A \in \mathcal{F}$) measurable. The unit sends x to δ_x , and the multiplication sends a measure on measures to its barycenter. Kleisli morphisms in the Giry monad are precisely Markov kernels — recovering the connection to Markov categories established in Chapter 6.

13.2 Kleisli Categories and Probabilistic Programs

Given a monad (T, η, μ) on $\mathcal{C}$, its *Kleisli category* $\mathcal{C}_T$ has the same objects as $\mathcal{C}$, and morphisms $A \rightarrow B$ in $\mathcal{C}_T$ are morphisms $A \rightarrow T(B)$ in $\mathcal{C}$ (“ T -computations” from A to B). Composition of $f : A \rightarrow T(B)$ and $g : B \rightarrow T(C)$ in $\mathcal{C}_T$ is:

$$g \circ_T f = \mu_C \circ T(g) \circ f : A \rightarrow T(C).$$

The identity at A in $\mathcal{C}_T$ is $\eta_A : A \rightarrow T(A)$.

For the probability monad, the Kleisli category has sets as objects and stochastic maps (functions into distributions) as morphisms; Kleisli composition is exactly the Chapman-Kolmogorov equation for composing stochastic kernels:

$$(g \circ_T f)(x)(C) = \int_{y \in B} g(y)(C) df(x)(y).$$

The Kleisli category of the Giry monad is precisely the category **Stoch** of Markov kernels. This shows that the Markov category framework of Chapter 6 arises from a monad, and that the two treatments are fully consistent.

Every probabilistic program — a program that samples from distributions, performs Bayesian conditioning, and returns a distribution over outputs — is a morphism in the Kleisli category of an appropriate probability monad. The categorical account of probabilistic programming (Staton, Heunen, Kammar et al.) is built on this observation.

13.3 The Writer and Reader Monads as Gradient Accumulators

Two monads from functional programming have direct interpretations in the context of automatic differentiation.

The *Writer monad* for a monoid $(M, \cdot, e)$ sends each object A to $A \times M$: a computation in the Writer monad is a computation that produces a value *together with* an element of M (a “log” or “accumulator”). The Kleisli maps $A \rightarrow B \times M$ represent functions that produce a value of type B and a logged element of M ; composition concatenates the logs using the

monoid multiplication.

In the context of forward-mode automatic differentiation, take $M = \mathbb{R}^n$ (the tangent space) with additive monoid structure. A Kleisli morphism $f : \mathbb{R}^m \rightarrow \mathbb{R}^n \times \mathbb{R}^n$ in the Writer monad is a function that computes both the value $f(x)$ and the Jacobian-vector product $Df(x) \cdot v$ (for a fixed tangent vector v). Kleisli composition accumulates the tangent vectors additively, which corresponds to the chain rule for forward-mode AD. The Writer monad thus provides the algebraic backbone of forward-mode differentiation.

The *Reader monad* for a fixed object E (the “environment”) sends A to the function object $[E, A]$: a computation in the Reader monad is a computation that reads from a fixed environment E . The Kleisli maps $A \rightarrow [E, B]$ represent functions that, given a value of type A and access to the environment, produce a value of type B .

In neural networks, the Reader monad arises naturally in the description of *hypernetworks*: networks whose weights are themselves the output of another network. The outer network reads from a “hyperparameter” environment and produces the weights for the inner network; this is exactly a Kleisli morphism in the Reader monad for the hyperparameter space as environment.

13.4 Comonads, Dropout, and Structured Context

A *comonad* (W, ε, δ) on $\mathcal{C}$ is the categorical dual of a monad: an endofunctor $W : \mathcal{C} \rightarrow \mathcal{C}$ with a *counit* $\varepsilon : W \Rightarrow \text{Id}_{\mathcal{C}}$ and a *comultiplication* $\delta : W \Rightarrow W \circ W$, satisfying the dual coassociativity and counit axioms. Dually, a comonad is a comonoid object in $[\mathcal{C}, \mathcal{C}]$.

The archetypal comonad in the context of spatial data is the *stream comonad*: on the category of sequences, $W(s)_n = (s_{n-k}, \dots, s_n, \dots, s_{n+k})$ (a sliding window of $2k + 1$ elements centred at position n). The counit $\varepsilon : W(s) \rightarrow s$ picks the central element s_n ; the comultiplication $\delta : W(s) \rightarrow W(W(s))$ applies the window operator twice, producing a window of windows. The Eilenberg-Moore algebras for the stream comonad are precisely the *translationally equivariant* maps — the maps that compute the same function at each position — which is the defining property of a convolutional filter.

In the probabilistic context, the *Bernoulli comonad* B_p on a suitable category of distributions (for a probability $p \in (0, 1)$) sends a distribution ν to the distribution $B_p(\nu)$ that, with probability p , is the original ν and, with probability $1 - p$, is a “zeroed out” version (corresponding to setting an activation to zero). The counit $\varepsilon : B_p \Rightarrow \text{Id}$ is the conditional expectation that recovers the original distribution; the comultiplication $\delta : B_p \Rightarrow B_p \circ B_p$ applies the mask twice independently.

This is, in essence, *dropout*: the regularisation technique of randomly zeroing out activations during training with probability p . The counit corresponds to the expectation over masks (the “prediction time” behaviour, where dropout is switched off and activations are scaled), and the comultiplication corresponds to the independence of the masks across layers. The comonad formulation of dropout predicts that dropout is compositionally consistent, because comonads compose: the composite of two comonadic “dropout” operations is again a comonadic operation, corresponding to applying dropout independently at two

different layers. This compositionality is empirically true and is the mathematical reason why dropout generalises well to deep networks.

13.5 The Eilenberg-Moore Category and Weight Sharing

The *Eilenberg-Moore category* $\mathcal{C}^T$ of a monad (T, η, μ) on $\mathcal{C}$ has as objects pairs (A, h) where $A \in \mathcal{C}$ and $h : T(A) \rightarrow A$ is a T -algebra structure map satisfying $h \circ \eta_A = \text{id}_A$ and $h \circ \mu_A = h \circ T(h)$. The morphisms are maps that commute with the algebra structure. There is a free-forgetful adjunction between $\mathcal{C}$ and $\mathcal{C}^T$ that generates the original monad.

Weight sharing in convolutional neural networks — the constraint that the same filter weights are applied at every spatial position — has a clean Eilenberg-Moore formulation. Consider the stream comonad W on the category of feature map sequences. The Eilenberg-Moore coalgebras (for the comonad dual) are the feature map sequences equipped with a fixed *response function*: a consistent rule for mapping each windowed neighbourhood to an output activation. Weight sharing is the constraint that this response function is given by a *fixed* filter applied uniformly across positions, and the condition that the convolution is equivariant with respect to translations is exactly the condition that the coalgebra structure map is a morphism in the Eilenberg-Moore category of the comonad. Different choices of comonad (sliding window of width k , atrous/dilated sliding window, pooling window) correspond to different convolutional architectures, all unified under the comonadic framework.

Chapter 14

Fibred Categories, Dependent Types, and Variable-Type Learning

14.1 Fibrations and Families of Categories

Many structures in mathematics and in machine learning involve a family of categories parametrised by another category: for each object B in a “base” category $\mathcal{B}$, there is a “fibre” category $\mathcal{E}_B$, and for each morphism $f : B \rightarrow B'$ in $\mathcal{B}$, there is a “reindexing functor” $f^* : \mathcal{E}_{B'} \rightarrow \mathcal{E}_B$. The correct categorical abstraction for this situation is a *Grothendieck fibration*.

Definition 14.1 (Grothendieck Fibration). A functor $p : \mathcal{E} \rightarrow \mathcal{B}$ is a *Grothendieck fibration* if for every morphism $f : B \rightarrow B'$ in $\mathcal{B}$ and every object $E' \in \mathcal{E}_{B'}$ (i.e., with $p(E') = B'$), there exists a *Cartesian lift*: a morphism $\bar{f} : \bar{E} \rightarrow E'$ in $\mathcal{E}$ with $p(\bar{f}) = f$ and the universal property that any other morphism in $\mathcal{E}$ over f factors uniquely through $\bar{f}$.

The universal property of Cartesian lifts ensures that the reindexing functors $f^* : \mathcal{E}_{B'} \rightarrow \mathcal{E}_B$ are well-defined and functorial: given $f : B \rightarrow B'$ and $g : B' \rightarrow B''$, the composite reindexing $(g \circ f)^*$ is naturally isomorphic to $f^* \circ g^*$. When $p : \mathcal{E} \rightarrow \mathcal{B}$ is a Grothendieck fibration, one speaks of $\mathcal{E}$ as a *$\mathcal{B}$ -indexed family of categories*.

The canonical example relevant to dependent types is the *codomain fibration* on **Set**: the functor $\text{cod} : \mathbf{Set}^{\rightarrow} \rightarrow \mathbf{Set}$ that sends each function $f : A \rightarrow B$ to its codomain B , and sends a commutative square to the morphism between codomains. The fibre over a set B is the slice category **Set**/ B (functions into B), and reindexing along $g : B \rightarrow B'$ is given by pulling back: $g^*(f : A \rightarrow B') = B \times_{B'} A \rightarrow B$.

In the context of dependent type theory, a fibration $p : \mathcal{E} \rightarrow \mathcal{B}$ is a model of a type theory in which types can *depend* on terms. The base category $\mathcal{B}$ is the category of contexts (tuples of variable declarations), the total category $\mathcal{E}$ is the category of types-in-context, and the fibration structure captures the substitution operation on types.

14.2 Dependent Optics and Variable-Type Learning

The optics of Chapter 4 were defined for a fixed pair of types (S, T) and (A, B) . In many learning scenarios, the type of the output depends on the input: the label of an image depends on the image itself (in a fine-grained classification task where the set of valid classes changes with the super-category), the answer to a question depends on the question (in VQA, different question types call for different answer formats). This type-dependence requires a generalisation of optics to the fibred setting, known as *dependent optics*.

A *dependent optic* from (S, T) to (A, B) — where B is a type family over A and T is a type family over S , both given as morphisms in a fibration $p : \mathcal{E} \rightarrow \mathcal{B}$ — is defined as an element of the coend:

$$\int^{M \in \mathcal{B}} \mathcal{B}(S, M \times A) \times_M \mathcal{E}_M(B_M, T_M),$$

where the fibre product $\times_M$ is taken over the common base object M , and $\mathcal{E}_M$ is the fibre category over M . The coend is taken in the 2-category of fibred categories, ensuring that the identification of residual objects is compatible with the fibration structure.

The dependent optic structure generalises the standard optic by allowing the residual state M to carry information about the type of the forward computation. In a dependent optic for a learning system, the forward pass computes both the intermediate state M and the (type-dependent) inner input A , while the backward pass maps the residual together with a type-dependent inner output back to the type T of the outer output. The type-dependence is critical: it allows the gradient to “know” the type of the prediction it is correcting.

An important special case is *masked language modelling*. Here the input S is a sequence of tokens (some masked), the type of the correct completion B depends on which tokens are masked (on $A = S$), and the model must predict the correct token at each masked position. The dependent optic structure captures the fact that the “type” of the prediction problem varies with the input: different masked sequences have different sets of positions to predict, and the gradient signal for each masked position has a type (a logit vector over the vocabulary at that position) that depends on the position’s identity.

14.3 The Sections Functor and Self-Supervised Learning

Given a fibration $p : \mathcal{E} \rightarrow \mathcal{B}$, a *section* of p is a functor $s : \mathcal{B} \rightarrow \mathcal{E}$ such that $p \circ s = \text{Id}_{\mathcal{B}}$: it assigns to each object B in the base an object $s(B)$ in the fibre $\mathcal{E}_B$, and to each morphism $f : B \rightarrow B'$ a morphism $s(f) : s(B) \rightarrow s(B')$ in $\mathcal{E}$ lying over f .

Self-supervised learning can be understood as the problem of finding a section of an appropriate fibration. Given a dataset of unlabelled inputs X and an augmentation fibration $p : \mathcal{E} \rightarrow \mathcal{B}$ (where $\mathcal{B}$ is the space of data points and $\mathcal{E}_x$ is the set of augmented views of x), a self-supervised representation is a section $s : \mathcal{B} \rightarrow \mathcal{E}$ that assigns to each data point its canonical representation — a representation that is invariant to augmentations within each

fibre. The contrastive loss (used in SimCLR, MoCo, CLIP) is the objective that measures how far the learned section is from being a true section of the augmentation fibration.

The *invariance condition* in contrastive learning — that representations of two augmented views of the same image should be close — is exactly the condition that the section is compatible with the fibration structure: $s(x) \cong s(x')$ whenever x and x' are in the same fibre (are augmentations of the same underlying image). The categorical formulation makes it clear that what is being learned is a structure-preserving map (a section), not an arbitrary function, and that the quality of the self-supervised representation is measured by how close it is to a genuine section of the fibration.

14.4 Slice Categories and Conditional Models

The *slice category* $\mathcal{C}/B$ over an object $B \in \mathcal{C}$ has as objects the morphisms $f : A \rightarrow B$ in $\mathcal{C}$ and as morphisms the commutative triangles over B . The “projection” functor $\mathcal{C}/B \rightarrow \mathcal{C}$ (sending $f : A \rightarrow B$ to A) is a Grothendieck fibration.

Conditional probability models $p(Y|X)$ live naturally in the slice category over X in the Markov category: a conditional model is a morphism in $\mathbf{Stoch}/X$ (a Markov kernel $X \rightarrow Y$, viewed as an object over X). The slice category over X parametrises all possible conditional distributions for a given fixed marginal over X , and the Bayesian inference operation (computing the posterior $p(X|Y)$ from the prior $p(X)$ and likelihood $p(Y|X)$) is a functor between slice categories, sending a conditional model $p(Y|X) \in \mathbf{Stoch}/X$ to the corresponding conditional $p(X|Y) \in \mathbf{Stoch}/Y$ via Bayes’ theorem.

This has immediate implications for the training of conditional models: the training objective (maximising $\log p(Y|X)$ over a dataset) is an optimisation problem in the slice category, and the gradient of the objective is a morphism in the tangent bundle fibration over $\mathbf{Stoch}/X$. The interaction between the Grothendieck fibration structure and the gradient structure gives a precise categorical account of the geometry of conditional model training, making it clear why the gradient of the log-likelihood is a tangent vector in the statistical manifold of the conditional model family.

Chapter 15

Double Categories, Traced Monoidal Structures, and Modular Architectures

15.1 Two Dimensions of Composition

An ordinary category has one dimension of composition: morphisms compose sequentially. A *double category* has *two* independent dimensions of composition, vertical and horizontal, that interact through an interchange law. This two-dimensional structure is precisely what is needed to describe systems that can be composed both sequentially (in time, as in the layers of a network) and in parallel (in space, as in the independent channels of a convolutional layer), with a coherent interaction between the two modes.

Definition 15.1 (Double Category). A (strict) *double category* $\mathbb{D}$ consists of:

1. a category $\mathbb{D}_0$ of *objects* and *vertical morphisms*,
2. a category $\mathbb{D}_1$ of *horizontal morphisms* and *2-cells* (squares),
3. source and target functors $s, t : \mathbb{D}_1 \rightarrow \mathbb{D}_0$, a unit functor $U : \mathbb{D}_0 \rightarrow \mathbb{D}_1$,
4. a horizontal composition functor $\odot : \mathbb{D}_1 \times_{\mathbb{D}_0} \mathbb{D}_1 \rightarrow \mathbb{D}_1$,

such that $(\mathbb{D}_0, \mathbb{D}_1, s, t, U, \odot)$ is a (strict) internal category in **Cat**.

The key structural element is the *2-cell*: given objects A, B, C, D , vertical morphisms $f : A \rightarrow C$ and $g : B \rightarrow D$, and horizontal morphisms $m : A \rightarrow B$ and $n : C \rightarrow D$, a 2-cell α fits into the square:

$$\begin{array}{ccc} A & \xrightarrow{m} & B \\ f \downarrow & \cong_{\alpha} & \downarrow g \\ C & \xrightarrow{n} & D \end{array}$$

Squares compose horizontally (side by side) and vertically (stacked), satisfying the interchange law: $(a \odot b) \cdot (c \odot d) = (a \cdot c) \odot (b \cdot d)$.

The lens category $\mathbf{Lens}(\mathcal{C})$ of Chapter 3 is the horizontal morphism category of a double category $\mathbb{L}$ in which vertical morphisms are reparametrisations (maps of state spaces) and 2-cells are pairs of vertical morphisms making the forward and backward passes commute. The double category structure captures the full algebraic coherence of the lens framework: vertical composition expresses how reparametrisations interact with the bidirectional structure, and the interchange law ensures that composing lenses and then reparametrising gives the same result as reparametrising and then composing. Gavranović’s doctoral thesis establishes this double-categorical framework as the correct setting for a rigorous treatment of the backpropagation functor.

15.2 Traced Monoidal Categories and Recurrence

A *traced symmetric monoidal category* (Joyal, Street, and Verity, 1996) is a symmetric monoidal category $(\mathcal{C}, \otimes, I, \sigma)$ equipped with a *trace* operation: for any morphism $f : A \otimes X \rightarrow B \otimes X$, a morphism $\mathrm{Tr}_{A,B}^X(f) : A \rightarrow B$, satisfying naturality in A , B , and X , the superposing axiom, the vanishing axiom (for the unit), and the yanking axiom $\mathrm{Tr}_{X,X}^X(\sigma_{X,X}) = \mathrm{id}_X$.

The trace formalises the “feedback” of the output X back into the input X : it represents a system that has an internal loop. In string diagrams, the trace corresponds to a wire that loops from an output port of f back to an input port of f , with the result being a morphism that no longer has the X wire visible externally.

For recurrent neural networks, the trace is the recurrence itself: the hidden state h is the object X that is “traced over”, and the RNN cell $f : X \otimes A \rightarrow B \otimes X$ (computing the next output B and next hidden state X from the current input A and current hidden state X) is the morphism being traced. The trace $\mathrm{Tr}^X(f) : A^\omega \rightarrow B^\omega$ (taken in a suitable category of streams) is the full unrolled computation of the RNN over an infinite sequence.

The yanking axiom says that tracing the identity morphism produces the identity: an RNN with no computation simply passes its hidden state unchanged. The naturality of the trace in X says that the result of the trace does not depend on the particular representation of the hidden state, only on its functional role. This is the categorical statement of the reparametrisation invariance of the RNN: two RNNs related by an invertible change-of-basis on their hidden states compute the same input-output function.

Theorem 15.2 (Int Construction: Joyal-Street-Verity, 1996). *Every traced symmetric monoidal category $(\mathcal{C}, \otimes, I, \sigma, \mathrm{Tr})$ embeds fully and faithfully into a compact closed category $\mathrm{Int}(\mathcal{C})$, in which the trace is realised by genuine bending of wires. The objects of $\mathrm{Int}(\mathcal{C})$ are pairs (A^+, A^-) of objects of $\mathcal{C}$ (the “positive” and “negative” wires), and morphisms $(A^+, A^-) \rightarrow (B^+, B^-)$ are morphisms $A^+ \otimes B^- \rightarrow B^+ \otimes A^-$ in $\mathcal{C}$.*

The Int construction shows that every recurrent computation can be embedded into a compact closed category, where the temporal recurrence of the RNN becomes spatial bending of wires. This provides a bridge between the theory of recurrent networks and the theory of compact closed categories developed in Chapter 2, and shows that the two frameworks are not separate but related by a canonical embedding.

15.3 Modular Architectures and Hypergraph Categories

A *modular architecture* is one in which sub-networks (modules) can be combined in flexible wiring topologies: the same module can appear multiple times (weight sharing), modules can be connected in any acyclic pattern (not just sequential), and the wiring can be data-dependent (conditional computation). The categorical framework for such architectures uses *hypergraph categories*, developed by Fong and Spivak.

A *hypergraph category* is a symmetric monoidal category in which every object A carries a special commutative Frobenius algebra structure: maps $\Delta : A \rightarrow A \otimes A$ (comultiplication), $\varepsilon : A \rightarrow I$ (counit), $\nabla : A \otimes A \rightarrow A$ (multiplication), and $\eta : I \rightarrow A$ (unit), satisfying the Frobenius condition $(\nabla \otimes \text{id}_A) \circ (\text{id}_A \otimes \Delta) = \Delta \circ \nabla = (\text{id}_A \otimes \nabla) \circ (\Delta \otimes \text{id}_A)$, as well as unitality and commutativity. The Frobenius condition is the key: it says that splitting and then merging wires is the same as merging and then splitting, and it allows arbitrary wiring topologies (not just trees) to be consistently defined.

Theorem 15.3 (Fong, 2016). *A category is a hypergraph category if and only if it is equivalent to a category of decorated corelations for some decoration functor. Decorated corelations are pairs (a corelation representing the wiring topology, a decoration assigning computations to each connected component of the wiring), and their composition is given by composing the corelations and restricting the decorations.*

For neural architectures, the decoration functor assigns to each wiring component (each wire or junction in the diagram) the type of data flowing through it, and to each computation box a parametrised function. The composition of decorated corelations corresponds to the assembly of modules into a network: connecting module outputs to module inputs, with the constraint that the types of connected ports must match. The hypergraph category structure guarantees that this assembly is consistent and associative — any two ways of assembling the same collection of modules with the same wiring produce the same overall computation.

15.4 The Calculus of Ends and Coends

Throughout this monograph, the constructions of optics, Tambara representations, and Kan extensions were stated using the language of ends and coends. We develop this language concisely here as a unified toolkit.

Let $F : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$ be a bifunctor. The *end* of F is an object $\int_{C \in \mathcal{C}} F(C, C) \in \mathcal{D}$ with projection morphisms $\omega_C : \int_{C \in \mathcal{C}} F(C, C) \rightarrow F(C, C)$ satisfying the *wedge condition*: for every $f : C \rightarrow C'$, $F(f, \text{id}_C) \circ \omega_C = F(\text{id}_{C'}, f) \circ \omega_{C'}$. The end picks out the “invariant” part of F under the diagonal action.

The *coend* $\int^{C \in \mathcal{C}} F(C, C)$ is the dual: an initial cowedge, amalgamating the values $F(C, C)$ while identifying elements related by the diagonal action. The foundational identity:

$$\text{Nat}(F, G) = \int_{C \in \mathcal{C}} \mathcal{D}(F(C), G(C))$$

shows that natural transformations are an end: the wedge condition is precisely naturality. The *Ninja Yoneda Lemma*:

$$\int^{C \in \mathcal{C}} \mathcal{C}(C, A) \times G(C) \cong G(A),$$

is the key tool for simplifying coend expressions, playing the role of β -reduction in the lambda calculus. Applied twice, it proves the Tambara representation theorem of Chapter 4. Profunctors $H : \mathcal{C} \rightharpoonup \mathcal{D}$ compose via the coend:

$$(K \circ H)(e, c) = \int^{d \in \mathcal{D}} K(e, d) \times H(d, c),$$

and the bicategory **Prof** of small categories and profunctors organises all hom-set-valued bifunctors into a single coherent structure. Attention in Transformers, viewed as the profunctor $(i, j) \mapsto A_{ij} \in [0, 1]$ between sequence positions, composes with the value assignment via exactly this coend formula, revealing that multi-layer attention is iterated profunctor composition.

Chapter 16

Higher Categories, Coherence, and the Future of the Theory

16.1 The Need for Higher Structure

Throughout this monograph we have worked with ordinary (1-)categories. In several places, however, we have encountered situations where morphisms themselves need to be related by “morphisms between morphisms”: the 2-cells of double categories, the natural transformations between natural transformations in functor categories, and the coherence isomorphisms of monoidal categories. When these higher-dimensional morphisms are themselves objects of study, we need the framework of *higher category theory*.

A *2-category* is a category enriched over **Cat**: it has objects, 1-cells (morphisms), and 2-cells (morphisms between morphisms), with two compositions (horizontal and vertical) satisfying the interchange law. A *bicategory* (or *weak 2-category*) weakens the associativity and unit laws to hold only up to specified 2-cell isomorphisms satisfying pentagon and triangle coherence. The bicategory **Prof** of profunctors is the canonical example: composition of profunctors is associative only up to natural isomorphism (the associator), and the coherence theorem guarantees that all these isomorphisms are compatible.

The hierarchy continues: a *tricategory* has 0-cells, 1-cells, 2-cells, and 3-cells, with the compositions satisfying coherence conditions proven by Gordon, Power, and Street. Beyond tricategories, the combinatorial complexity of specifying coherence laws grows rapidly, and the correct approach is through the theory of $(\infty, 1)$ -categories.

16.2 $(\infty, 1)$ -Categories and Homotopy-Coherent Backpropagation

An $(\infty, 1)$ -category has k -morphisms for all $k \geq 0$, with the k -morphisms for $k \geq 2$ all being invertible (equivalences). The correct implementation-independent way to define such structures is through *quasi-categories* (Joyal, Lurie): simplicial sets X satisfying the inner horn-filling condition, where every inner horn $\Lambda_k^n \rightarrow X$ (for $0 < k < n$) has a filler

$\Delta^n \rightarrow X$. The n -simplices of X are the n -morphisms.

Lurie’s monumental treatises *Higher Topos Theory* (2009) and *Higher Algebra* (2017) develop the full machinery of $(\infty, 1)$ -categories: limits, colimits, adjunctions, monads, operads, and symmetric monoidal structures, all in the homotopy-coherent setting. The key point is that in an $(\infty, 1)$ -category, every statement about equality is replaced by a statement about contractible spaces of equivalences: there are not just finitely many coherence conditions (as in the pentagon for monoidal categories), but an infinite tower of homotopy-coherent conditions, all of which are satisfied automatically in the quasi-categorical framework.

For deep learning, the relevant $(\infty, 1)$ -categorical structures are the following. The Parametrised construction, properly formulated, should be a morphism in the $(\infty, 1)$ -category of parametrised smooth maps, where the equivalences between parametrisations (reparametrisations) are themselves coherently tracked at all levels. The backpropagation functor BP of Chapter 3, in this setting, becomes an $(\infty, 1)$ -functor: it sends k -morphisms (homotopies between reparametrisations) to k -morphisms (homotopies between lens reparametrisations) for all k , and all these maps are compatible.

The practical significance of the $(\infty, 1)$ -categorical formulation is that it provides, automatically and without additional verification, the full tower of coherence conditions for any categorical construction involving parametrised maps. A framework for deep learning architecture design built on $(\infty, 1)$ -categorical foundations would be, by construction, free of all coherence bugs: any two ways of composing the same collection of layers, with the same wiring, would produce provably equivalent computations, at all levels of homotopical detail.

16.3 Homotopy Type Theory and Formal Verification

Homotopy type theory (HoTT), presented in the *Homotopy Type Theory* book (the Univalent Foundations Program, 2013), is a formal system that integrates the $(\infty, 1)$ -categorical structure into the foundations of mathematics. In HoTT, every type has a homotopy type: the identity types $\text{Id}_A(a, b)$ form a space (an ∞ -groupoid) that captures all the higher-dimensional equivalences between elements. The *Univalence Axiom* (Voevodsky) states that equivalence of types is the same as identity: $(A \simeq B) \simeq (A = B)$, making structural (categorical) equivalence a formal equality.

For deep learning, the HoTT perspective suggests a programme of formal verification: expressing the properties of a learning algorithm (correctness of backpropagation, compositional generalisation, robustness bounds, equivariance) as types in HoTT, and constructing formal proofs (elements of those types) that the algorithm has those properties. The Univalence Axiom guarantees that any property that holds for one representation of a network also holds for any isomorphic (reparametrised) representation — the precise formal content of representation invariance.

Early steps in this direction have been taken: the categorical structures developed in this monograph (lenses, optics, Markov categories) have formal type-theoretic definitions in the

Agda and Coq proof assistants, and the lens composition formula (which is the heart of the correctness of backpropagation) has been formally verified. The full formalisation of the backpropagation functor BP in HoTT, including its monoidal structure, remains an open problem of significant technical difficulty and practical importance.

16.4 Coherence as a Compiler Guarantee

A leitmotif of this monograph has been *coherence*: the guarantee that all ways of composing the basic operations of a mathematical structure give the same result. Mac Lane’s coherence theorem for monoidal categories (any two ways of reassociating a tensor product are equal, via the canonical associator), the Joyal-Street theorem for string diagrams (any topologically equivalent string diagrams compute the same morphism), and the Tambara representation theorem for optics (any two representations of an optic via different residual types are naturally isomorphic) are all coherence theorems. So is the correctness of the chain rule in the CRDC framework, and the associativity of lens composition.

From the perspective of deep learning engineering, coherence theorems are *compiler guarantees*: they ensure that the modular decomposition of a network into components, the choice of implementation for each component, and the assembly of components into a full network all produce the same result, regardless of the order in which these choices are made. A framework for deep learning that is built on coherent categorical structures — one in which every operation is a morphism in an appropriately structured category, and the categorical axioms are satisfied — automatically comes with these guarantees, without requiring ad hoc verification for each specific network.

The contrast with current practice is stark. Modern deep learning frameworks (PyTorch, JAX, TensorFlow) provide *some* compositional guarantees: automatic differentiation is correct for any computation graph expressible in the framework’s language, and this guarantee follows from the chain rule (the CRDC axiom, in categorical language). But these guarantees are proved for specific primitives and specific composition patterns, and the verification that a novel architectural feature (a new normalisation scheme, a new attention variant, a new positional encoding) interacts correctly with the AD system and the distributed training infrastructure requires ad hoc analysis.

A framework built on the categorical foundations developed in this monograph would, by construction, have coherence guarantees for all compositions of all operations, making it substantially easier to reason formally about the correctness of complex architectures. The string diagram calculus would serve as a graphical programming language for neural architectures, with the Joyal-Street theorem guaranteeing that any topologically equivalent diagrams produce equal computations, and the coherence theorems for monoidal categories, traced categories, and compact closed categories providing the full toolkit of formal manipulations.

16.5 Open Problems and Future Directions

We close by identifying the open problems that, in our view, are the most important for the further development of the categorical theory of deep learning.

The *homotopy-coherent Para construction* is the problem of making the Para construction of Chapter 3 fully $(\infty, 1)$ -categorical: defining a quasi-category $\mathbf{Para}_\infty(\mathcal{C})$ for a suitable $(\infty, 1)$ -category $\mathcal{C}$, and showing that the backpropagation functor is an $(\infty, 1)$ -functor from $\mathbf{Para}_\infty(\mathbf{Smooth})$ to $\mathbf{Lens}_\infty(\mathbf{Smooth})$. This requires, in particular, a homotopy-coherent version of the CRDC axioms, which does not yet exist in the literature.

The *categorical treatment of normalisation layers* is the problem of giving a precise categorical account of BatchNorm, LayerNorm, GroupNorm, and their variants. These operations introduce a global statistical dependence (across batch or sequence dimensions) that breaks the simple sequential composition structure of feedforward networks: the normalisation at one layer depends on all inputs in the batch, not just the current one. The correct categorical framework for this would involve some notion of *dependent* composition, likely in the fibred category setting of Chapter 14.

The *categorical semantics of prompting and in-context learning* is the problem of giving a categorical account of the phenomenon by which a large language model can learn new tasks from a small number of examples presented in the input context. This is not learning in the parametric sense (the weights do not change), but a form of computation that resembles learning. The relationship between prompting and the Para construction (where the “parameters” are the in-context examples rather than the weights) and between in-context learning and the Kan extension formulation of generalisation (Chapter 12) is suggestive but not yet fully worked out.

The *categorical theory of scaling laws* is the problem of giving a categorical account of the empirical observation that the performance of large language models scales as a power law in the number of parameters, compute, and data. The scaling law is a statement about the relationship between the size of the Para object (the parameter space) and the quality of the best-case morphism in $\mathbf{Para}(\mathbf{Smooth})$ that fits a given dataset. The categorical framework should provide a way of making this statement precise and of deriving it from structural properties of the category of smooth maps and the dataset distribution.

Finally, the *formal verification of transformer correctness* is the problem of providing a machine-checked proof, in a formal system (Agda, Lean, Coq), that the Transformer architecture correctly implements the bidirectional computation described by its optic structure, that the attention mechanism satisfies the appropriate equivariance conditions (invariance to permutation of key-value pairs), and that the positional encoding satisfies the appropriate coherence conditions with respect to the sequence ordering. This is a well-posed mathematical problem, and its solution would provide the strongest possible formal guarantee of correctness for the most widely used architecture in deep learning.

Chapter 17

Operads, Multicategories, and the Algebra of Network Wiring

17.1 Beyond Binary Composition

The categorical structures developed in earlier chapters rely on a single binary composition: morphisms $f : A \rightarrow B$ and $g : B \rightarrow C$ combine into $g \circ f : A \rightarrow C$. Many operations in neural networks, however, are fundamentally *multilinear*: the attention mechanism takes three inputs (query, key, value) and produces one output; a bilinear layer takes two inputs and produces one; a loss function takes a prediction and a label and produces a scalar. The correct categorical abstraction for operations with multiple inputs of potentially different types is not an ordinary category but a *multicategory* or, when the inputs are unordered, an *operad*.

Definition 17.1 (Multicategory). A *multicategory* $\mathcal{M}$ consists of: a collection of objects; for each finite list of objects $A_1, \dots, A_n$ and object B , a set $\mathcal{M}(A_1, \dots, A_n; B)$ of *multimorphisms* (or *operations*); identity morphisms $\text{id}_A \in \mathcal{M}(A; A)$; and a *substitution* (or *multilinear composition*) operation: given $f \in \mathcal{M}(A_1, \dots, A_n; B)$ and $g_i \in \mathcal{M}(B_{i,1}, \dots, B_{i,k_i}; A_i)$ for each i , a composite $f \circ (g_1, \dots, g_n) \in \mathcal{M}(B_{1,1}, \dots, B_{n,k_n}; B)$; satisfying associativity and unit laws for substitution.

An ordinary category is a multicategory in which all multimorphisms have arity exactly one ($n = 1$). A *symmetric multicategory* additionally equips each $\mathcal{M}(A_1, \dots, A_n; B)$ with an action of the symmetric group Σ_n by permutation of inputs, compatible with composition.

An *operad* is a symmetric multicategory with a single object. The hom-sets $\mathcal{O}(n) = \mathcal{M}(*, \dots, *; *)$ (n inputs, one output, all of the unique type $*$) are the sets of n -ary operations, and the composition of an n -ary operation with n operations of arities $k_1, \dots, k_n$ is the substitution. The operad structure captures the compositional structure of a class of operations, without regard to what those operations act on.

17.2 Algebras over Operads and Neural Architecture Families

Given an operad $\mathcal{O}$ and a symmetric monoidal category $(\mathcal{C}, \otimes, I)$, an *algebra over $\mathcal{O}$ in $\mathcal{C}$* is an object $A \in \mathcal{C}$ together with a collection of morphisms $\theta_n : \mathcal{O}(n) \otimes A^{\otimes n} \rightarrow A$ (for each $n \geq 0$) satisfying equivariance with respect to the Σ_n -action on $\mathcal{O}(n)$ and compatibility with the operadic composition. An algebra over $\mathcal{O}$ in **Set** is a set equipped with one n -ary operation for each element of $\mathcal{O}(n)$, satisfying the corresponding relations.

The relevance to neural networks is the following. Consider the *associative operad* **Ass**, whose n -ary operations are the elements of the symmetric group Σ_n (representing the $n!$ ways of composing n inputs in sequence). Algebras over **Ass** in **Set** are precisely monoids. In **FVect** $_{\mathbb{R}}$, algebras over **Ass** are associative algebras. In the category of neural network layers, algebras over **Ass** are *sequential* architectures: families of layers that can be composed in any order, with the composition operation given by the sequential composition of function application.

The *commutative operad* **Comm**, whose n -ary operations form a single element (the unique commutative n -ary operation), gives algebras that are commutative monoids (**Set**), commutative algebras (**FVect** $_{\mathbb{R}}$), or *permutation-invariant* architectures in the neural setting: architectures like DeepSets and set-input networks that treat all inputs symmetrically.

The *little n -disks operad* $\mathcal{D}_n$, due to Boardman and Vogt and central to the study of iterated loop spaces in algebraic topology, captures operations that can be performed at different locations in n -dimensional space and composed by spatial inclusion. Algebras over $\mathcal{D}_n$ are E_n -algebras, and for $n = 1$ these are homotopy-associative (but not necessarily commutative) structures. The relevance to attention-based architectures is that self-attention can be understood as performing operations (reading and writing) at different positions in a sequence, and the operadic structure of the little intervals operad $\mathcal{D}_1$ captures the way in which these operations can be composed along the sequence dimension.

17.3 The Operadic Composition of Attention Heads

The multi-head attention mechanism of the Transformer can be given a precise operadic formulation. Recall that multi-head attention with H heads computes:

$$\text{MHA}(Q, K, V) = \text{Concat}(h_1, \dots, h_H)W^O,$$

where $h_i = \text{Attn}(QW_i^Q, KW_i^K, VW_i^V)$ is the i -th attention head and W^O is an output projection. Each head is a morphism $h_i : (\mathbb{R}^{n \times d})^{\times 3} \rightarrow \mathbb{R}^{n \times d_v}$ (from the triple of query, key, value matrices to a head output), and the overall multi-head attention is a morphism $\text{MHA} : (\mathbb{R}^{n \times d})^{\times 3} \rightarrow \mathbb{R}^{n \times d}$.

The operadic structure is the following. Define a *symmetric operad* $\mathcal{A}$ of attention operations, where $\mathcal{A}(3)$ is the set of all single-head attention operations (parametrised by query, key, and value projection matrices), $\mathcal{A}(H \cdot 3)$ is the set of all multi-head attention operations with H heads, and the operadic composition is given by head concatenation

followed by output projection. The algebra over $\mathcal{A}$ in the category of sequence-to-sequence functions is the Transformer block.

This operadic perspective has two consequences. First, it gives a precise notion of *operadic composition of attention*: the output of one multi-head attention module can serve as the input to the next, and the operadic composition law guarantees that this is consistent with the individual head computations. Second, it provides a natural notion of *head pruning*: removing heads from a multi-head attention module corresponds to restricting the operad $\mathcal{A}(H \cdot 3)$ to a sub-operad with fewer heads, and the consistency of this restriction is a consequence of the operadic structure.

17.4 Coloured Operads and Typed Wiring Diagrams

A *coloured operad* (or *symmetric multicategory*) generalises the single-object operad by allowing operations with inputs of different types. Given a set of colours C , a C -coloured operad has sets of operations $\mathcal{O}(c_1, \dots, c_n; d)$ for each list of input colours $c_1, \dots, c_n$ and output colour d . The composition substitutes an operation f of output colour c into an input slot of colour c in another operation.

In neural network terms, colours are *tensor types*: the shape and dtype of tensors. A coloured operad of neural network operations has one colour for each tensor type (e.g., “float32 tensor of shape $[B, n, d]$ ”), and the operations are the neural network layers that accept inputs of specified shapes and produce outputs of specified shapes. The operadic composition is the assembly of layers into a network, with the type-checking of tensor shapes enforced by the colour-matching condition: an output of colour c can only be plugged into an input slot of colour c .

This is essentially what a strongly-typed computational graph framework implements: the colours are the dtype-shape pairs, the operadic composition is the graph construction, and type errors (mismatched tensor shapes) correspond to violations of the colour-matching condition. The operadic formulation shows that this type-checking is not merely a programming convenience but a mathematical necessity: it is the condition for the composition of network operations to be well-defined in the coloured operad.

Spivak’s *wiring diagrams* (developed in the context of systems theory and operadic composition) provide a graphical calculus for coloured operads that generalises the string diagrams of Chapter 2 to the multi-input setting. A wiring diagram is a graph in which boxes represent operations and wires represent the flow of data between them, with the colour-matching condition enforced by type labels on the wires. The composition of wiring diagrams is the substitution of one diagram into a box of another, and the resulting diagram correctly computes the composite operation.

17.5 The Boardman-Vogt Tensor Product and Interchange

Given two operads $\mathcal{O}$ and $\mathcal{P}$, their *Boardman-Vogt tensor product* $\mathcal{O} \otimes_{\text{BV}} \mathcal{P}$ is the operad in which operations of $\mathcal{O}$ and $\mathcal{P}$ can be applied in any order, with the interchange law

governing their interaction: applying an $\mathcal{O}$ -operation and then a $\mathcal{P}$ -operation gives the same result as applying the $\mathcal{P}$ -operation and then the $\mathcal{O}$ -operation, up to a specified homotopy. Algebras over $\mathcal{O} \otimes_{\mathbf{BV}} \mathcal{P}$ are structures that are simultaneously algebras over $\mathcal{O}$ and over $\mathcal{P}$, with the two algebra structures interchangeable.

For neural networks, the most important case is $\mathcal{D}_1 \otimes_{\mathbf{BV}} \mathcal{D}_1 \simeq \mathcal{D}_2$ (a theorem in algebraic topology): the tensor product of two copies of the little intervals operad is equivalent to the little disks operad in two dimensions. This means that a structure that is separately “sequential” along two independent dimensions (e.g., along the sequence dimension and along the batch dimension) is automatically an algebra over the two-dimensional disks operad, which imposes commutativity conditions in the two-dimensional sense. The relevance to Transformer architectures, which operate along both the sequence and the head dimension, and to vision Transformers (ViT), which operate along two spatial dimensions, is immediate.

Chapter 18

Equivariant Architectures and Categorical Symmetry

18.1 Symmetry in Machine Learning

The guiding principle of geometric deep learning (Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2017; Bronstein, Bruna, Cohen, Veličković, 2021) is that effective neural network architectures should be designed to respect the symmetries of the data. A function $f : X \rightarrow Y$ between spaces equipped with group actions of a group G is *G -equivariant* if $f(g \cdot x) = g \cdot f(x)$ for all $g \in G$ and all $x \in X$, and *G -invariant* if $f(g \cdot x) = f(x)$. Convolutional neural networks are translation-equivariant (the translation group $\mathbb{Z}^2$ acts on images by shifting); graph neural networks are permutation-equivariant (the symmetric group Σ_n acts on graphs by relabelling nodes); point cloud networks should be rotation-equivariant (the special orthogonal group $SO(3)$ acts on point clouds).

The categorical framework makes the symmetry structure of a network architecture an intrinsic part of the morphism, not an additional property to be verified. We develop this in full generality.

18.2 Group Actions and the Category of G -Spaces

Fix a group G (or more generally a groupoid, or a monoid, but we treat the group case for concreteness). A *G -set* is a set X equipped with a left action $\cdot : G \times X \rightarrow X$ satisfying $e \cdot x = x$ and $(gh) \cdot x = g \cdot (h \cdot x)$. The category $G\text{-}\mathbf{Set}$ of G -sets and G -equivariant maps is equivalent to the functor category $[\mathbf{BG}, \mathbf{Set}]$, where $\mathbf{BG}$ is the one-object category with morphisms G (the *delooping* of G): a functor $F : \mathbf{BG} \rightarrow \mathbf{Set}$ assigns to the unique object $*$ a set $F(*) = X$ and to each group element $g \in G$ a function $F(g) : X \rightarrow X$ satisfying $F(e) = \text{id}_X$ and $F(gh) = F(g) \circ F(h)$, which is precisely a G -action.

This identification — $G\text{-}\mathbf{Set} \cong [\mathbf{BG}, \mathbf{Set}]$ — is the key: it means that the theory of G -sets and equivariant maps is a special case of functor category theory. Natural transformations between G -set functors are equivariant maps, and the Yoneda lemma for $[\mathbf{BG}, \mathbf{Set}]$ specialises to the *orbit-stabiliser theorem*: the hom-set $\text{Hom}_{G\text{-}\mathbf{Set}}(G/H, X)$ (equivariant

maps from the orbit space G/H to X) is in natural bijection with the fixed point set $X^H = \{x \in X : h \cdot x = x \text{ for all } h \in H\}$.

In the context of representation learning: the Yoneda embedding of a G -set X into $[\mathbf{BG}, \mathbf{Set}]^{\text{op}}$ is the “equivariant Yoneda representation”, and a G -equivariant neural network is a natural transformation in $[\mathbf{BG}, \mathcal{C}]$ for some appropriate enriched category $\mathcal{C}$.

18.3 Linear Representations and Schur's Lemma Categorically

A *linear representation* of G is a G -action on a vector space V : a group homomorphism $\rho : G \rightarrow GL(V)$. The category $\mathbf{Rep}_k(G)$ of k -linear representations of G (with G -equivariant linear maps as morphisms) is equivalent to the functor category $[\mathbf{BG}, \mathbf{Vect}_k]$. It is abelian (morphisms have kernels and cokernels), and when G is finite and $\text{char}(k) = 0$, it is semisimple (Maschke's theorem): every representation decomposes as a direct sum of irreducibles.

Theorem 18.1 (Schur's Lemma, Categorically). *In the functor category $[\mathbf{BG}, \mathbf{Vect}_k]$, the hom-space between two irreducible representations V and W is:*

$$\text{Hom}_G(V, W) \cong \begin{cases} k & \text{if } V \cong W, \\ 0 & \text{if } V \not\cong W. \end{cases}$$

Categorically: the irreducible representations are the simple objects of the abelian category $\mathbf{Rep}_k(G)$, and Schur's lemma is the statement that the only endomorphisms of a simple object are scalar multiples of the identity.

For equivariant neural networks, the decomposition of the representation space into irreducibles is the analogue of frequency decomposition in harmonic analysis. An equivariant linear layer $f : V \rightarrow W$ (a G -equivariant linear map) is, by Schur's lemma, block-diagonal in the basis of irreducibles: the map from the ρ -isotypic component of V (the direct sum of all copies of the irreducible ρ in V) to the ρ -isotypic component of W is a scalar (if the multiplicities are both 1) or a matrix of scalars (if the multiplicities are greater). This is the mathematical content of the “weight sharing” in equivariant networks: Schur's lemma forces the weight matrices to respect the symmetry, and the effective number of free parameters is the number of pairs of matching irreducible components.

18.4 The Category of Equivariant Neural Networks

Define the category $\mathbf{EqNN}_G$ of G -equivariant neural networks as follows. Objects are pairs (V, ρ) where V is a finite-dimensional real vector space and $\rho : G \rightarrow GL(V)$ is a representation. Morphisms $(V, \rho) \rightarrow (W, \sigma)$ are G -equivariant smooth maps $f : V \rightarrow W$ satisfying $f(\rho(g)v) = \sigma(g)f(v)$ for all $g \in G$ and $v \in V$. Composition is composition of equivariant maps.

This category is a full subcategory of $[\mathbf{BG}, \mathbf{Smooth}]$, the functor category of smooth

G -spaces and equivariant smooth maps. The inclusion functor $\mathbf{EqNN}_G \hookrightarrow \mathbf{Smooth}$ (forgetting equivariance) has a right adjoint only when G is compact (by averaging over the group to produce a G -equivariant approximation to any smooth map), which is the categorical expression of the fact that equivariant approximation by averaging is possible only for compact groups.

An equivariant neural network is a morphism in $\mathbf{Para}(\mathbf{EqNN}_G)$: a G -equivariant parametrised map. The parameter space P carries a trivial G -action (parameters are not transformed by G), and the implementation map $I : P \times V \rightarrow W$ must be G -equivariant in its second argument: $I(p, \rho(g)v) = \sigma(g)I(p, v)$ for all p, g, v . The backpropagation functor $\mathbf{BP} : \mathbf{Para}(\mathbf{EqNN}_G) \rightarrow \mathbf{Lens}(\mathbf{EqNN}_G)$ restricts correctly to the equivariant setting: the gradient of an equivariant loss with respect to an equivariant network is equivariant, and the backward pass preserves the G -equivariance of the representations.

18.5 Group Convolution as a Natural Transformation

The *group convolution* of a signal $f : G \rightarrow \mathbb{R}$ with a filter $\psi : G \rightarrow \mathbb{R}$ is:

$$(f \star \psi)(g) = \sum_{h \in G} f(h)\psi(h^{-1}g) = \sum_{h \in G} f(gh^{-1})\psi(h).$$

The standard convolution on $\mathbb{Z}^2$ (for images) is the special case $G = \mathbb{Z}^2$ with the translation action on $\ell^2(\mathbb{Z}^2)$. Group convolution is G -equivariant by construction: $(L_g f) \star \psi = L_g(f \star \psi)$, where $L_g f(h) = f(g^{-1}h)$ is the left regular action.

Categorically, group convolution is a natural transformation between the left-regular representation and itself:

$$\mathbf{Conv}_\psi : \ell^2(G) \rightarrow \ell^2(G), \quad f \mapsto f \star \psi,$$

and naturality is exactly G -equivariance. The Fourier transform on a compact group G (or on $\mathbb{Z}^2$ for discrete images) diagonalises the convolution:

$$\widehat{f \star \psi} = \hat{f} \cdot \hat{\psi},$$

which is a natural isomorphism between the convolution functor (from $\ell^2(G)$ to $\ell^2(G)$ as G -representations) and a pointwise multiplication functor on the Fourier side. This natural isomorphism is the categorical content of the convolution theorem.

Theorem 18.2 (Peter-Weyl Theorem, Categorically). *For a compact group G , the regular representation $L^2(G)$ decomposes as:*

$$L^2(G) \cong \bigoplus_{\pi \in \hat{G}} V_\pi \otimes V_\pi^*,$$

where $\hat{G}$ is the set of isomorphism classes of irreducible unitary representations π of G , and V_π is the representation space. Categorically, this is the statement that the functor $\mathbf{ev}_* : \mathbf{Rep}(G) \rightarrow \mathbf{Set}$ (evaluation at the unit element $*$ of G) is monadic, and the monad is exactly the “convolution with the regular representation” monad.

For equivariant convolutional networks (Cohen-Welling, 2016; Cohen et al., 2019), the Peter-Weyl decomposition determines the correct weight-sharing pattern. Each equivariant layer between feature maps of types $V = \bigoplus_{\pi} m_{\pi} V_{\pi}$ and $W = \bigoplus_{\sigma} n_{\sigma} V_{\sigma}$ has a weight matrix that is block-diagonal in the irreducible basis, with each block being an $n_{\sigma} \times m_{\pi}$ matrix of scalars (by Schur’s lemma). The number of free parameters is $\sum_{\pi=\sigma} n_{\pi} \cdot m_{\pi}$, which is dramatically smaller than the $(\dim W) \times (\dim V)$ parameters of a general linear layer.

18.6 Steerable CNNs and Fibre Bundles

Steerable CNNs (Cohen and Welling, 2017; Weiler et al., 2018) generalise equivariant convolutions to the setting of feature maps that are not scalar-valued but *vector-valued*: each spatial position carries a vector that transforms under a representation of the group. The mathematical structure is a *vector bundle* over the base space (the spatial grid), equipped with a G -action that simultaneously moves the basepoint and transforms the fibre.

Categorically, steerable CNNs are morphisms in the category of *equivariant vector bundles*: objects are G -equivariant vector bundles $E \rightarrow X$ over a G -space X (where G acts simultaneously on the base and the fibres), and morphisms are G -equivariant vector bundle maps. The convolution operation on steerable feature maps is a morphism in this category, and its G -equivariance is automatic from the morphism condition.

The Grothendieck fibration of Chapter 14 appears here in a concrete form: the “total space” of equivariant feature maps fibres over the base space X (the spatial domain), and the fibre over each point $x \in X$ is the representation space $\rho(G_x)$ (the representation of the stabiliser of x). A steerable filter is a section of an associated bundle, and the convolution of a feature map with a filter is the pushforward of this section along the G -action.

Chapter 19

Information Geometry and the Natural Gradient Categorically

19.1 Statistical Manifolds and the Fisher Metric

Information geometry (Amari, 1985; Amari-Nagaoka, 2000) is the study of families of probability distributions as Riemannian manifolds, with the Riemannian metric given by the *Fisher information matrix*. For a parametric family $\{p_\theta : \theta \in \Theta \subseteq \mathbb{R}^d\}$ of probability distributions on a measurable space X , the Fisher information matrix is:

$$\mathcal{I}(\theta)_{ij} = \mathbb{E}_{x \sim p_\theta} \left[\frac{\partial \log p_\theta(x)}{\partial \theta_i} \frac{\partial \log p_\theta(x)}{\partial \theta_j} \right] = -\mathbb{E}_{x \sim p_\theta} \left[\frac{\partial^2 \log p_\theta(x)}{\partial \theta_i \partial \theta_j} \right].$$

The matrix $\mathcal{I}(\theta)$ is positive semi-definite and defines a Riemannian metric on Θ (the *Fisher-Rao metric*), making $(\Theta, \mathcal{I})$ a Riemannian manifold called the *statistical manifold*.

The categorical formulation of information geometry begins with the observation that the space of probability distributions on X is not merely a set but an object in the Markov category **BorelStoch**. The statistical manifold Θ is (when smooth) a smooth manifold, and the map $\theta \mapsto p_\theta$ is a smooth embedding $\iota : \Theta \hookrightarrow \mathcal{G}(X)$ (into the Giry monad applied to X). The Fisher metric is the pullback of the canonical Riemannian metric on $\mathcal{G}(X)$ (the L^2 metric on square-root densities, also known as the *Hellinger metric*) along this embedding:

$$\mathcal{I}(\theta) = \iota^* g_{\mathcal{G}(X)},$$

where $g_{\mathcal{G}(X)}$ is the Fisher-Rao metric on the full space of distributions.

19.2 Amari's Dual Connections

The Fisher-Rao metric on a statistical manifold comes with a family of torsion-free affine connections, parametrised by $\alpha \in \mathbb{R}$: the α -connections of Amari. The two most important are the *exponential connection* $\nabla^{(+1)}$ and the *mixture connection* $\nabla^{(-1)}$, which are *dual*

with respect to the Fisher metric in the sense that:

$$X\langle Y, Z \rangle = \langle \nabla_X^{(+1)} Y, Z \rangle + \langle Y, \nabla_X^{(-1)} Z \rangle$$

for all vector fields X, Y, Z on Θ , where $\langle \cdot, \cdot \rangle$ is the Fisher metric. A Riemannian manifold with two dual connections is called a *statistical manifold* in the sense of Lauritzen. When both connections are flat (zero curvature), the manifold is a *dually flat space*, and Amari shows that all exponential families are dually flat.

The categorical interpretation of the dual connections was identified by Ay, Jost, Lê, and Schwachhöfer (2017) and by Loaiza-Ganem and Cunningham (2019): the exponential connection corresponds to the *pushforward* of distributions along smooth maps (the covariant direction), while the mixture connection corresponds to the *pullback* (the contravariant direction). In the language of the Grothendieck fibration of Chapter 14, the total space of the statistical tangent bundle fibres over the statistical manifold, with the two connections corresponding to the two fundamental directions of the fibration structure.

More precisely, in the Markov category framework, there is a functor:

$$\mathcal{I} : \mathbf{Smooth}^{\text{op}} \rightarrow \mathbf{Riem}$$

from the opposite of the category of smooth manifolds and smooth maps to the category of Riemannian manifolds and isometric embeddings, sending each smooth manifold M to the statistical manifold of smooth probability distributions on M , and each smooth map $f : M \rightarrow N$ to the induced embedding of statistical manifolds. The Fisher information is a natural transformation from this functor to the “scalar curvature” functor, expressing the fact that the Fisher information is an intrinsic invariant of the statistical manifold, not depending on any particular coordinate system.

19.3 The Natural Gradient as a Riemannian Gradient

The *natural gradient* (Amari, 1998) replaces the Euclidean gradient in the parameter update rule with the Riemannian gradient on the statistical manifold:

$$\tilde{\nabla}_{\theta} \mathcal{L} = \mathcal{I}(\theta)^{-1} \nabla_{\theta} \mathcal{L}.$$

The motivation is that the Euclidean gradient does not account for the curvature of the statistical manifold: a step of size ε in parameter space corresponds to a step of variable size in distribution space, depending on the local curvature $\mathcal{I}(\theta)$. The natural gradient is the gradient that takes a step of uniform size (in the Fisher-Rao metric) in distribution space.

Categorically, the natural gradient is the component of the gradient in the *cotangent bundle* of the statistical manifold, viewed as a Riemannian manifold. The Fisher metric provides an isomorphism $\flat : T\Theta \rightarrow T^*\Theta$ between the tangent and cotangent bundles (musical isomorphisms), and its inverse $\sharp = \mathcal{I}^{-1} : T^*\Theta \rightarrow T\Theta$ sends the cotangent gradient $d\mathcal{L} \in T^*\Theta$ to the Riemannian gradient $\tilde{\nabla} \mathcal{L} = \mathcal{I}^{-1} d\mathcal{L} \in T\Theta$.

The backpropagation functor $\mathbf{BP}$ of Chapter 3 computes the cotangent gradient $d\mathcal{L}$ (the usual gradient, which lives in the cotangent space); the natural gradient update requires the additional step of applying $\mathcal{I}^{-1}$. This additional step is a morphism in the tangent bundle category, and the natural gradient descent algorithm is a morphism in $\mathbf{Para}(\mathbf{Smooth})$ that factors through the Fisher metric.

Proposition 19.1. *The natural gradient descent update $\theta \mapsto \theta - \eta \mathcal{I}(\theta)^{-1} \nabla_{\theta} \mathcal{L}$ is the gradient descent in the Riemannian metric $\mathcal{I}$, and it is the morphism in $\mathbf{Para}(\mathbf{Smooth})$ obtained by composing the backpropagation lens with the musical isomorphism $\sharp : T^*\Theta \rightarrow T\Theta$ induced by the Fisher metric. The invariance of the natural gradient under reparametrisation of the statistical manifold is a consequence of the naturality of the musical isomorphism: it commutes with the action of diffeomorphisms on the tangent and cotangent bundles.*

The proof is a direct computation: under a reparametrisation $\phi : \Theta \rightarrow \Phi$ (a diffeomorphism), the gradient transforms as $\nabla_{\phi} = (D\phi)^{-\top} \nabla_{\theta}$ (contravariant), the Fisher metric transforms as $\mathcal{I}_{\phi} = (D\phi)^{-\top} \mathcal{I}_{\theta} (D\phi)^{-1}$ (as a $(0, 2)$ -tensor), and their combination $\mathcal{I}_{\phi}^{-1} \nabla_{\phi} = D\phi \mathcal{I}_{\theta}^{-1} \nabla_{\theta}$ is covariant (transforms as a vector). The natural gradient update is therefore a well-defined vector field on the statistical manifold, independent of the parametrisation.

19.4 Categorical KL Divergence and the Information-Geometric Adjunction

The *KL divergence* $\mathrm{KL}(p\|q) = \int_X p(x) \log(p(x)/q(x)) dx$ is the canonical “distance” measure on the space of probability distributions, though it is not a true distance (it is not symmetric and does not satisfy the triangle inequality). In the Markov category framework, the KL divergence has a categorical interpretation as the *log-likelihood ratio* of two morphisms $p, q : I \rightarrow \mathcal{G}(X)$ in $\mathbf{BorelStoch}$.

More precisely, the *reverse KL* $\mathrm{KL}(q\|p)$ (used in variational inference) and the *forward KL* $\mathrm{KL}(p\|q)$ (used in maximum likelihood estimation) are two different divergences, corresponding to the two dual connections on the statistical manifold. The *information-geometric adjunction* is the statement that m -projection onto an exponential family minimises the forward KL, while e -projection minimises the reverse KL, and these two projections are adjoint in the appropriate categorical sense.

Formally, consider the inclusion functor $\iota : \mathcal{E} \hookrightarrow \mathcal{G}(X)$ of an exponential family $\mathcal{E}$ into the full space of distributions. The *m-projection* $\Pi_m : \mathcal{G}(X) \rightarrow \mathcal{E}$ (if it exists) is the right adjoint to ι with respect to the mixture connection: $\Pi_m(q) = \arg \min_{p \in \mathcal{E}} \mathrm{KL}(p\|q)$. The *e-projection* $\Pi_e : \mathcal{G}(X) \rightarrow \mathcal{E}$ is the right adjoint with respect to the exponential connection: $\Pi_e(q) = \arg \min_{p \in \mathcal{E}} \mathrm{KL}(q\|p)$. The adjunction $\iota \dashv \Pi_m$ in the dually flat geometry is Amari’s information-geometric generalisation of the projection theorem in Hilbert spaces.

This adjunction underlies the EM algorithm: the E-step computes the m -projection of the full posterior onto the variational family, and the M-step maximises the expected log-likelihood (which is the e -projection in the exponential family). The alternating optimisation of EM is the categorical alternation of the two adjoint projections.

19.5 The Categorical Cramér-Rao Bound

The *Cramér-Rao bound* is the fundamental inequality of statistical estimation theory: for any unbiased estimator $\hat{\theta}$ of the parameter θ of a statistical model p_θ , the covariance of $\hat{\theta}$ is at least $\mathcal{I}(\theta)^{-1}$:

$$\text{Cov}_\theta(\hat{\theta}) \succeq \mathcal{I}(\theta)^{-1}.$$

This is a lower bound on the variance of any estimator, and it is tight (achieved by the maximum likelihood estimator in the limit of large samples, under regularity conditions).

The categorical formulation of the Cramér-Rao bound uses the *variance functor* $\text{Var} : \mathbf{Stoch} \rightarrow \mathbf{PosDef}$ (from Markov kernels to positive definite matrices, well-defined for Gaussian families) and the *Fisher information functor* $\mathcal{I} : \mathbf{Smooth} \rightarrow \mathbf{PosDef}$ (from smooth statistical models to positive definite matrices). The Cramér-Rao bound is the statement that the natural transformation $\text{Var} \Rightarrow \mathcal{I}^{-1}$ (expressing the bound) is well-defined and its components are positive semi-definite. This natural transformation is the *data processing inequality* applied to the specific case of estimation: processing data through an estimator can only reduce information, and the Fisher information quantifies how much information is available.

In the context of deep learning, the Cramér-Rao bound applies to the precision of gradient estimation: when gradients are estimated from mini-batches (as in stochastic gradient descent), the variance of the gradient estimate is bounded below by the Fisher information of the loss function with respect to the mini-batch distribution. Networks that approximate the natural gradient (like K-FAC, EKFAC, and Shampoo) are precisely computing better approximations to the inverse Fisher metric, reducing the variance of the gradient estimate toward its Cramér-Rao lower bound.

Chapter 20

Categorical Signal Processing: Convolution, the Fourier Transform, and Spectral Methods

20.1 Convolution as a Morphism in a Monoidal Category

The convolution of two functions $f, g : \mathbb{R} \rightarrow \mathbb{R}$ (or $\mathbb{Z} \rightarrow \mathbb{R}$, for discrete signals) is:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(s)g(t-s) ds.$$

This operation is bilinear, commutative ($f \star g = g \star f$), and associative: $(f \star g) \star h = f \star (g \star h)$. The space $L^1(\mathbb{R})$ with convolution as multiplication is a commutative Banach algebra, and the Fourier transform is a ring homomorphism from this algebra to the algebra of continuous functions on the frequency domain.

The categorical account of convolution begins with the observation that $L^1(\mathbb{R})$ (or $\ell^1(\mathbb{Z})$ for discrete signals) is a *monoid object* in the monoidal category $(\mathbf{Ban}, \hat{\otimes}, \mathbb{C})$ of Banach spaces with the projective tensor product $\hat{\otimes}$. Convolution is the monoid multiplication, commutativity of convolution corresponds to commutativity of the monoid, and associativity is the monoid associativity axiom. The shift operator $T_s : f(t) \mapsto f(t-s)$ is the right action of the additive group $(\mathbb{R}, +)$ on $L^1(\mathbb{R})$, making convolution a G -equivariant operation for $G = (\mathbb{R}, +)$.

For neural networks, the discrete version is most relevant. The space $\ell^2(\mathbb{Z}^2)$ of square-summable sequences indexed by $\mathbb{Z}^2$ (representing images as biinfinite arrays) is a Hilbert space, and convolution with a fixed filter $\psi \in \ell^1(\mathbb{Z}^2)$ is a bounded linear operator $C_\psi : \ell^2(\mathbb{Z}^2) \rightarrow \ell^2(\mathbb{Z}^2)$, $C_\psi(f) = f \star \psi$. The key property is that C_ψ is a *translation-equivariant* operator: it commutes with all shifts $T_{(m,n)}(f)(x,y) = f(x-m, y-n)$. By the Fourier characterisation of translation-equivariant operators, *every* bounded translation-equivariant linear operator on $\ell^2(\mathbb{Z}^2)$ is a convolution operator — this is the theorem that motivates the convolutional architecture.

Theorem 20.1 (Characterisation of Equivariant Linear Operators). *A bounded linear operator $L : \ell^2(\mathbb{Z}^d) \rightarrow \ell^2(\mathbb{Z}^d)$ commutes with all translations (is $\mathbb{Z}^d$ -equivariant) if and only if it is a convolution operator $L = C_\psi$ for some $\psi \in \ell^\infty(\mathbb{Z}^d)$. In categorical terms: the full subcategory of $\mathbf{B}(\ell^2(\mathbb{Z}^d))$ (bounded operators) consisting of $\mathbb{Z}^d$ -equivariant operators is equivalent to the algebra of Fourier multipliers on $\ell^\infty(\mathbb{Z}^d)$.*

This theorem is the foundational justification for convolutional neural networks: if we require the weight-sharing property (equivariance under translation), then the only possible linear operations are convolutions. The non-linear activation functions (ReLU, GELU, etc.) break the linearity but preserve equivariance (since $\sigma(\rho(g)v) = \rho(g)\sigma(v)$ when σ is applied elementwise and $\rho(g)$ is a permutation of coordinates).

20.2 The Fourier Transform as a Monoidal Functor

The Fourier transform $\mathcal{F} : L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R})$, $\hat{f}(\xi) = \int_{\mathbb{R}} f(t)e^{-2\pi i \xi t} dt$, is an isometric isomorphism of Hilbert spaces (Plancherel's theorem). In the categorical framework, it is far more: it is a *monoidal functor*.

Consider the symmetric monoidal category $(L^1(\mathbb{R}), \star, \delta_0)$, where objects are L^1 functions, the monoidal product is convolution $\star$, and the unit is the Dirac delta δ_0 (satisfying $f \star \delta_0 = f$). On the frequency side, consider the symmetric monoidal category $(\mathcal{C}_0(\mathbb{R}), \cdot, 1)$, where objects are continuous functions vanishing at infinity, the monoidal product is pointwise multiplication $\cdot$, and the unit is the constant function 1.

Theorem 20.2 (Fourier Transform as Monoidal Functor). *The Fourier transform $\mathcal{F} : (L^1(\mathbb{R}), \star, \delta_0) \rightarrow (\mathcal{C}_0(\mathbb{R}), \cdot, 1)$ is a symmetric monoidal functor: it sends the monoidal unit δ_0 to the unit 1 and preserves the monoidal product: $\mathcal{F}(f \star g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$ (the convolution theorem).*

The naturality of the Fourier transform with respect to the group structure of $\mathbb{R}$ is the statement that $\mathcal{F} \circ T_s = e^{-2\pi i s \cdot} \mathcal{F}$ (a shift in time corresponds to a phase shift in frequency), which is a natural isomorphism between the functors “shift by s and then Fourier transform” and “Fourier transform and then multiply by $e^{-2\pi i s \cdot}$ ”. This naturality is the Fourier shift theorem, and it is what makes the Fourier transform the canonical tool for analysing translation-equivariant operations.

The monoidal functor perspective immediately gives the correct framework for spectral convolution in graph neural networks (GCNs). For a graph $G = (V, E)$ with Laplacian $L = D - A$ (where D is the degree matrix and A is the adjacency matrix), the graph Fourier transform $\hat{f} = U^\top f$ (where U is the matrix of eigenvectors of L) diagonalises the graph convolution: $f \star_G g = U(U^\top f \odot U^\top g)$. This is the monoidal functor $\mathcal{F}_G : (\mathbb{R}^{|V|}, \star_G, \delta) \rightarrow (\mathbb{R}^{|V|}, \odot, \mathbf{1})$, and the graph convolution theorem is the statement that $\mathcal{F}_G(f \star_G g) = \mathcal{F}_G(f) \odot \mathcal{F}_G(g)$.

20.3 Spectral Graph Theory and the Laplacian Functor

The graph Laplacian $L = D - A$ is a canonical object associated to a graph, and it is the key operator in spectral graph theory and spectral GCNs. In categorical terms, the Laplacian is a natural transformation:

$$\mathcal{L} : \mathbf{Grph} \rightarrow \mathbf{LinOp}$$

from the category of graphs and graph morphisms to the category of linear operators (on vector spaces indexed by the vertex set). Naturality means that if $\phi : G \rightarrow H$ is a graph morphism, then the Laplacians L_G and L_H are intertwined: $L_H \circ \phi_* = \phi_* \circ L_G$ (where $\phi_* : \mathbb{R}^{V(G)} \rightarrow \mathbb{R}^{V(H)}$ is the pushforward along ϕ). This is exactly the equivariance of the Laplacian under graph isomorphisms.

Message passing in graph neural networks (Gilmer et al., 2017) can be expressed as the iteration of a natural transformation $\mathbf{MP} : \mathbf{Grph}_{\text{feat}} \rightarrow \mathbf{Grph}_{\text{feat}}$ on the category of graphs with vertex features. The message passing update rule:

$$h_v^{(k+1)} = \phi \left(h_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} \psi(h_v^{(k)}, h_u^{(k)}, e_{vu}) \right)$$

is a natural transformation because it is defined intrinsically in terms of the graph structure (neighbourhood relations, edge features) and does not depend on any labelling of the vertices. The permutation equivariance of GNNs — the fact that relabelling the vertices does not change the output — is exactly the naturality of $\mathbf{MP}$.

The Weisfeiler-Leman hierarchy of graph isomorphism tests (1-WL, 2-WL, k -WL) can be rephrased as a hierarchy of natural transformations: the k -WL test is (essentially) the natural transformation induced by the functor $\text{Hom}(-, K_k) : \mathbf{Grph}^{\text{op}} \rightarrow \mathbf{Set}$ (homomorphisms from the complete graph K_k), and the separation power of the k -WL test is determined by whether this functor can distinguish non-isomorphic graphs. The categorical reformulation gives a clean proof of the known equivalences between GNN expressive power and WL hierarchy levels.

20.4 Wavelets and the Scattering Transform as a Functor

The *scattering transform* (Mallat, 2012) is an architecture for signal processing that is translation-invariant and deformation-stable, defined by alternating wavelet convolution and modulus nonlinearity:

$$S_J[q]f = |W_J| * |W_{j_1}| * \cdots * |W_{j_n}|f$$

for multi-index $q = (j_1, \dots, j_n)$ with $j_k < J$. Here W_j is a wavelet filter at scale j , and $|W_j| * f$ means convolve with the wavelet and take the modulus. The scattering transform is stable under the action of diffeomorphisms: if $\tau : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a C^1 diffeomorphism close to a translation, then $\|Sf - S(f \circ \tau)\| \leq C \|\nabla \tau\|_\infty \|f\|_{L^2}$.

Categorically, the scattering transform is a functor $S_J : (\mathbf{Diff}^1, \|\cdot\|_\infty, \mathbb{R}^d) \rightarrow (L^2, \|\cdot\|, \mathbb{R})$ from the metric space of C^1 -diffeomorphisms (with the C^1 -norm) to the metric space of L^2 feature maps (with the L^2 norm), satisfying the *enriched functor* condition (non-expansiveness up to the constant C). The stability bound is exactly the Lipschitz constant of the functor in the enriched-category sense of Chapter 7.

This gives a categorical explanation of why the scattering transform is useful for audio and image classification: it is a Lipschitz functor from the space of natural signals (viewed as a metric space with the deformation metric) to a feature space, and Lipschitz functors preserve the “hardness” of the classification problem in a controlled way. Any classifier on top of the scattering features has error that differs from the optimal error on the original signals by at most a Lipschitz factor.

Chapter 21

The Free Energy Principle, Active Inference, and Categorical Cybernetics of Self-Organisation

21.1 The Free Energy Principle

The *Free Energy Principle* (FEP), developed by Karl Friston and collaborators over the past two decades, proposes that all self-organising biological systems minimise a quantity called *variational free energy*, which is an upper bound on the *surprisal* (negative log evidence) of their sensory states under an implicit generative model of the world. The FEP is a unified framework for perception, action, learning, and attention, and it has attracted significant interest from the machine learning community because it subsumes deep learning, reinforcement learning, and Bayesian brain theories as special cases.

The variational free energy is defined, for a generative model $p(o, s)$ (over observations o and latent states s) and a variational distribution $q(s)$ (the “recognition model” or “posterior approximation”), as:

$$F(o, q) = \underbrace{-\mathbb{E}_{q(s)}[\log p(o, s)] + \mathbb{E}_{q(s)}[\log q(s)]}_{\text{energy} - \text{entropy}} = \underbrace{-\log p(o) + \text{KL}(q(s) \| p(s|o))}_{\text{surprisal} + \text{KL}}.$$

Minimising F with respect to q (holding o fixed) minimises the KL divergence from q to the true posterior $p(s|o)$, which is exactly variational Bayes or the VAE objective of Chapter 6. Minimising F with respect to actions (which affect o) drives the agent to take actions that make its observations more predictable under its generative model — active inference.

21.2 Categorical Formulation of Active Inference

The categorical account of active inference was developed by Sajid, Parr, Friston et al. (2021) and by Capucci, Gavranović, Hedges, and Rischel (2022), building on the categorical cybernetics framework of Chapter 10. The key insight is that the free energy functional

$F : \mathcal{G}(S) \times \mathcal{O} \rightarrow \mathbb{R}$ (from pairs of recognition models and observations to free energy values) is a lens in the Markov category.

Specifically, the active inference loop can be described as a lens $\ell_{\text{FEP}} : (Q, \mathcal{O}) \rightarrow (Q, \mathcal{A})$ where:

- $Q = \mathcal{G}(S)$ is the space of recognition distributions (variational posteriors over latent states S),
- $\mathcal{O}$ is the observation space,
- $\mathcal{A}$ is the action space,
- $\text{get}(q, o) =$ the optimal action under the free energy gradient: the action that minimises F given current beliefs q and observation o ,
- $\text{put}(q, o, \delta a) =$ the updated belief state after observing the result of action a in the world.

The composition of this lens with the environment lens $\ell_{\text{env}} : (\mathcal{A}, S) \rightarrow (\mathcal{A}, \mathcal{O})$ (describing how the environment maps actions to observations) gives the full perception-action loop.

Theorem 21.1 (Capucci-Gavranović-Hedges-Rischel, 2022). *The active inference loop is a fixed point of the composition:*

$$\ell_{\text{FEP}} \circ \ell_{\text{env}} : (Q \times \mathcal{A} \times S, Q \times \mathcal{O}) \rightarrow (Q \times \mathcal{A} \times S, Q \times \mathcal{O}),$$

*in the lens category **Lens(Stoch)**. This fixed point is exactly the condition that the agent's beliefs q are consistent with the observations o generated by the environment under the agent's actions, and the actions are optimal given those beliefs — the definition of active inference equilibrium.*

The formal proof uses the Eilenberg-Moore category of the Giry monad (Chapter 13) to identify the fixed points of the composition with the solutions of a Bayesian update equation, and the lens composition formula (Chapter 3) to show that these fixed points satisfy the free energy minimisation condition.

21.3 The Markov Blanket and Category-Theoretic Self-Organisation

The *Markov blanket* of a set of states μ in a Bayesian network is the minimal set $b(\mu)$ of states that renders μ conditionally independent of all other states: given $b(\mu)$, the states μ are independent of the environment. In active inference, the Markov blanket separates the internal states of the agent (brain) from the external states of the world, with the blanket consisting of sensory and active states at the boundary.

The categorical formulation of the Markov blanket uses the *conditional independence structure* of Markov categories developed in Chapter 6. The Markov blanket condition is:

$$p(\mu, \eta | b) = p(\mu | b)p(\eta | b),$$

which in Markov category terms is the statement that the joint morphism $p : I \rightarrow \mathcal{G}(\mu \times \eta \times b)$ factors (over b) as:

$$I \xrightarrow{p_b} \mathcal{G}(b) \xrightarrow{\delta_b} \mathcal{G}(b) \otimes \mathcal{G}(b) \xrightarrow{p_{\mu|b} \otimes p_{\eta|b}} \mathcal{G}(\mu) \otimes \mathcal{G}(\eta)$$

where δ_b is the copy morphism of the Markov category. The category-theoretic formulation makes it clear that the Markov blanket is a *cospan* in the Markov category: a span of conditional distributions with the blanket states b as the apex.

The FEP then states that self-organising systems act to *maintain* their Markov blanket: they take actions to ensure that the blanket states remain in a low-entropy configuration (consistent with the generative model), which is the same as minimising the variational free energy restricted to the blanket states. The categorical formulation reveals that this is a *fixed-point condition* in the Markov category: the system is at equilibrium when its marginal distribution over blanket states is a fixed point of the Markov blanket update morphism.

21.4 Connections to Predictive Coding and Deep Learning

Predictive coding (Rao and Ballard, 1999; Clark, 2013) is the neuroscientific theory that the brain implements a hierarchical generative model, where each level of the hierarchy predicts the activity of the level below and updates its predictions based on prediction errors. The mathematical structure of predictive coding is a hierarchical application of variational Bayes, and it has inspired the development of biologically plausible learning algorithms that can be implemented without backpropagation.

The categorical formulation of predictive coding is a nested sequence of lenses:

$$\ell_1 \circ \ell_2 \circ \cdots \circ \ell_L : (Q_1 \times \cdots \times Q_L, \mathcal{O}) \rightarrow (Q_1 \times \cdots \times Q_L, \mathcal{O}),$$

where $\ell_k : (Q_k, E_{k-1}) \rightarrow (Q_k, E_k)$ is the lens at level k (with E_k the prediction error at level k , and Q_k the recognition distribution at level k). The composition of lenses computes the full predictive coding update, and the fixed point of the full composition is the equilibrium of the predictive coding network.

The connection to the backpropagation functor of Chapter 3 is as follows. The forward pass of a deep network (computing the prediction $\hat{y}$ from the input x) corresponds to the “downward” pass of the predictive coding network (generating a prediction at each level from the beliefs at that level). The backward pass (computing the gradient of the loss with respect to the weights) corresponds to the “upward” pass of the predictive coding network (propagating prediction errors from the bottom level to the top). The backpropagation algorithm and the predictive coding update are thus two presentations of the same lens composition, viewed from opposite ends of the hierarchy.

This categorical equivalence between backpropagation and predictive coding was conjectured by Hinton (2022) and has been partially proved in specific settings. The lens-category formulation provides the precise categorical statement: the two algorithms define the same

morphism in **Lens(Stoch)**, and their equivalence is a theorem about the composition of lenses, not a statement about any particular implementation.

21.5 Categorical Thermodynamics and the Free Energy Landscape

The free energy landscape of a parametric model p_θ is the function $\theta \mapsto F(\theta) = -\log p_\theta(o) + \text{KL}(q_\phi \| p_\theta(s|o))$, which depends on both the generative parameters θ and the recognition parameters ϕ . Gradient descent on the free energy landscape (with respect to both θ and ϕ) is the variational EM algorithm, which alternates between optimising ϕ (E-step, updating the recognition model) and optimising θ (M-step, updating the generative model).

The categorical thermodynamics framework of Baez and Fong (2018) and Loaiza-Ganem et al. provides a precise categorical account of the relationship between the free energy landscape and the thermodynamic free energy of a physical system. The key observation is that the variational free energy $F(o, q) = \mathbb{E}_q[\log q(s)] - \mathbb{E}_q[\log p(o, s)]$ is the sum of the *entropy* $H(q) = -\mathbb{E}_q[\log q(s)]$ (corresponding to the thermodynamic entropy of the system) and the *expected energy* $U(o, q) = -\mathbb{E}_q[\log p(o, s)]$ (corresponding to the thermodynamic internal energy). The Legendre transform of the free energy with respect to the “temperature” parameter β gives the partition function $Z(\beta, o) = \int e^{-\beta F(o, q)} dq$, which is a natural transformation in the Markov category.

The categorical framework makes the following precise: the free energy functional F is a *colax monoidal functor* from the category of generative models and Markov kernels to the poset $(\mathbb{R}, \geq)$ (ordered by the free energy value). Colax monoidality means that $F(p \otimes p') \geq F(p) + F(p')$ for independent systems, which is the *subadditivity* of free energy — a fundamental property that guarantees the non-trivial content of the FEP as a physical principle rather than a tautology.

Chapter 22

Loss Functions, Divergences, and the Geometry of Learning Objectives

22.1 Loss Functions as Morphisms

A *loss function* in supervised learning is a map $\ell : \hat{Y} \times Y \rightarrow [0, \infty)$ from pairs (prediction, target) to non-negative reals. The most common examples are the squared loss $\ell_2(\hat{y}, y) = \|\hat{y} - y\|^2$, the cross-entropy loss $\ell_{\text{CE}}(\hat{p}, y) = -\log \hat{p}(y)$, and the hinge loss $\ell_{\text{hinge}}(\hat{y}, y) = \max(0, 1 - y\hat{y})$. Each of these is a morphism in a suitable category.

In **Smooth** (smooth maps between manifolds), the squared loss is a morphism $\ell_2 : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, \infty)$, and the cross-entropy loss is a morphism $\ell_{\text{CE}} : \Delta^{n-1} \times \{1, \dots, n\} \rightarrow [0, \infty)$ (from the probability simplex crossed with the label set). In the Markov category **BorelStoch**, the cross-entropy is a Markov kernel $\ell_{\text{CE}} : \Delta^{n-1} \times \{1, \dots, n\} \rightarrow \mathcal{G}([0, \infty))$ (which assigns to each prediction-label pair a distribution over loss values, a generalisation needed for stochastic loss functions).

The categorical study of loss functions asks: what properties of a loss function ensure that minimising it produces a good predictor? The answer, in categorical terms, is *properness*: a loss function ℓ is *proper* if for every distribution p over labels Y , the prediction $\hat{y}^* = \arg \min_{\hat{y}} \mathbb{E}_{y \sim p}[\ell(\hat{y}, y)]$ is the distribution p itself (the optimal prediction under ℓ is the true conditional distribution). Proper loss functions include the squared loss, cross-entropy, and log loss; the hinge loss is not proper.

Theorem 22.1 (Savage, 1971; Gneiting-Raftery, 2007). *A loss function $\ell : \mathcal{P}(Y) \times Y \rightarrow [0, \infty)$ is proper if and only if there exists a convex function $G : \mathcal{P}(Y) \rightarrow \mathbb{R}$ (the Bayes risk or entropy function) such that $\ell(p, y) = G(p) - \nabla G(p) \cdot (e_y - p)$, where e_y is the one-hot encoding of y . The loss is strictly proper if and only if G is strictly convex.*

The categorical interpretation is: proper loss functions are in bijection with convex functions on the probability simplex, and the bijection is given by the *Fenchel-Moreau dual* (the Legendre transform). The Bayes risk G is the concave function $G(p) = -\mathbb{E}_{y \sim p}[\ell(p^*, y)]$ (the minimum expected loss over all predictors), and its Legendre transform $G^*(z) = \sup_p [z \cdot p - G(p)]$ gives the individual loss values.

In categorical terms, the correspondence between proper losses and convex functions is a natural isomorphism of functors:

$$\mathbf{ProperLoss}(Y) \cong \mathbf{Conv}(\mathcal{P}(Y)),$$

where $\mathbf{ProperLoss}(Y)$ is the set of proper loss functions on label space Y and $\mathbf{Conv}(\mathcal{P}(Y))$ is the set of continuous convex functions on the probability simplex, both viewed as functors on the category of label spaces and measurable maps. Naturality means that the correspondence commutes with measurable relabelling of the label space, which is the statement that properness is an intrinsic property (not depending on the labelling convention).

22.2 Bregman Divergences and Dually Flat Geometry

A *Bregman divergence* for a convex function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ is:

$$D_\phi(x||y) = \phi(x) - \phi(y) - \nabla\phi(y) \cdot (x - y).$$

The squared loss is the Bregman divergence for $\phi(x) = \|x\|^2$; the KL divergence is the Bregman divergence for $\phi(p) = \sum_i p_i \log p_i$ (negative entropy) on the probability simplex; the Itakura-Saito divergence (used in audio processing) is the Bregman divergence for $\phi(x) = -\log x$.

Bregman divergences have three key properties: (1) $D_\phi(x||y) \geq 0$ with equality iff $x = y$ (non-negativity and definiteness); (2) they are not symmetric in general ($D_\phi(x||y) \neq D_\phi(y||x)$); (3) they satisfy a generalised Pythagorean theorem: if $z^* = \arg \min_{z \in C} D_\phi(z||y)$ is the Bregman projection of y onto a convex set C , then $D_\phi(x||y) = D_\phi(x||z^*) + D_\phi(z^*||y)$ for all $x \in C$.

The categorical interpretation of Bregman divergences uses the dually flat geometry of information geometry (Chapter 19). Every Bregman divergence defines a dually flat space: the primal coordinates θ (natural parameters) and the dual coordinates $\eta = \nabla\phi(\theta)$ (expectation parameters) are related by the Legendre transform ϕ^* , and the two coordinate systems correspond to the two dual connections $\nabla^{(+1)}$ and $\nabla^{(-1)}$. The generalised Pythagorean theorem is the statement that the e -geodesic and the m -geodesic from z^* to y are orthogonal with respect to the Fisher metric, and its categorical proof uses the orthogonality of the two dual connections.

Theorem 22.2 (Banerjee et al., 2005). *There is a bijection between proper loss functions on a finite label space Y and Bregman divergences on the probability simplex $\mathcal{P}(Y)$: the proper loss ℓ corresponds to the Bregman divergence $D_{G^*}(p||q) = G^*(p) - G^*(q) - \nabla G^*(q) \cdot (p - q)$ (where G^* is the Legendre transform of the Bayes risk G). This bijection is a natural isomorphism:*

$$\mathbf{ProperLoss}(Y) \cong \mathbf{Bregman}(\mathcal{P}(Y)).$$

In categorical terms, this theorem says that the two categories $\mathbf{ProperLoss}(Y)$ and $\mathbf{Bregman}(\mathcal{P}(Y))$ are equivalent (in fact, in bijection), and the bijection is a natural isomorphism with respect to the action of measurable maps on the label space. It gives a

complete classification of “reasonable” loss functions in terms of the convex analysis of the probability simplex.

22.3 f -Divergences and the Data Processing Inequality

An f -divergence (Ali-Silvey, 1966; Csiszár, 1967) is defined, for a convex function $f : (0, \infty) \rightarrow \mathbb{R}$ with $f(1) = 0$, as:

$$D_f(p\|q) = \int_X q(x) f\left(\frac{p(x)}{q(x)}\right) dx.$$

Special cases include: $f(t) = t \log t$ gives the KL divergence; $f(t) = -\log t$ gives the reverse KL; $f(t) = (t - 1)^2$ gives the χ^2 -divergence; $f(t) = (\sqrt{t} - 1)^2$ gives the squared Hellinger distance.

The key property of f -divergences is the *data processing inequality*: for any Markov kernel $K : X \rightarrow \mathcal{P}(Y)$ and any two distributions p, q on X :

$$D_f(K_*p\|K_*q) \leq D_f(p\|q),$$

where $K_*p = \int_X K(y|x)p(x)dx$ is the pushforward. Processing data through a channel can only reduce (or maintain) the f -divergence.

The categorical account of the data processing inequality is the statement that every f -divergence is a *colax monoidal* natural transformation from the functor $\mathcal{G}(-)^{\times 2}$ (pairs of distributions) to the functor $(-)_+$ (non-negative reals), where the monoidal structure is given by Markov kernel composition. Colax monoidality means $D_f(K_*(p, q)) \leq D_f(p, q)$ for all Markov kernels K , which is the data processing inequality.

Theorem 22.3 (Choi-Yamanaka-Katayama, 2017; Category-Theoretic Form). *f -divergences are precisely the natural transformations:*

$$D : \mathbf{BorelStoch}(-, X)^{\times 2} \Rightarrow [0, \infty]$$

that are: (1) non-negative and vanish on the diagonal; (2) lower semicontinuous; (3) jointly convex; (4) satisfy the data processing inequality. The f -divergence corresponding to a convex function f is the unique natural transformation satisfying these conditions and $D(p_\theta\|p_{\theta'}) = D_f(p_\theta\|p_{\theta'})$ for all Gaussian pairs $(p_\theta, p_{\theta'})$.

The theorem classifies all “reasonable” divergence measures as f -divergences, and identifies each with a convex function f via the Gaussians as a “test family”. The categorical statement is a characterisation theorem: the f -divergences are the natural transformations from the product presheaf to the real line, satisfying a short list of categorical conditions.

22.4 PAC-Bayes Bounds as Categorical Inequalities

PAC-Bayes theory (Shawe-Taylor and Williamson, 1997; McAllester, 1999) provides generalisation bounds for stochastic predictors. Given a prior P on the hypothesis space $\mathcal{H}$, a

posterior Q (possibly depending on the training data), and a training set S of m examples, the PAC-Bayes bound states: with probability $1 - \delta$ over the draw of S ,

$$\mathbb{E}_{h \sim Q}[\text{err}(h)] \leq \mathbb{E}_{h \sim Q}[\text{er}(h, S)] + \sqrt{\frac{\text{KL}(Q \| P) + \log(m/\delta)}{2m}},$$

where $\text{err}(h)$ is the true error and $\text{er}(h, S)$ is the empirical error.

The categorical interpretation of the PAC-Bayes bound uses the Markov category formalism. The bound is an inequality between two functors on **BorelStoch**: the “expected true error” functor $\mathcal{E} : \mathcal{G}(\mathcal{H}) \rightarrow [0, 1]$ and the “empirical error plus complexity penalty” functor $\hat{\mathcal{E}} : \mathcal{G}(\mathcal{H}) \times \mathcal{H}^m \rightarrow [0, \infty)$. The bound is a *natural inequality* (a lax natural transformation) between these functors, holding with high probability over the draw of training data.

The KL divergence term $\text{KL}(Q \| P)$ that appears in the bound is precisely the measure of how far the posterior Q is from the prior P — the “information cost” of the posterior given the prior, in the information-geometric sense. The categorical reformulation makes it clear that the PAC-Bayes bound is a combination of two categorical ingredients: the concentration of measure inequality (which gives the high-probability statement about the training data) and the data processing inequality (which ensures that the KL divergence correctly measures the information cost of the posterior).

The optimised PAC-Bayes bound (choosing the posterior Q to minimise the bound) gives the *Gibbs posterior*:

$$Q^*(h) \propto P(h)e^{-\beta \text{er}(h, S)},$$

which is the Boltzmann distribution with energy er and inverse temperature β . Categorically, the Gibbs posterior is the unique posterior that satisfies the fixed-point condition of the Bayesian update monad (Chapter 13): it is the fixed point of the Kleisli morphism that maps the prior P to the posterior Q^* via Bayes’ rule with the empirical likelihood $e^{-\beta \text{er}}$.

Chapter 23

Generative Models: GANs, Flows, and Diffusion as Categorical Structures

23.1 The Category of Generative Models

A generative model is, at its most abstract, a morphism $g : Z \rightarrow X$ in some category, where Z is a *latent space* and X is the data space. The task of generative modelling is to find a morphism g such that the pushforward $g_*\mu_Z$ of a simple reference distribution μ_Z on Z (typically a standard Gaussian $\mathcal{N}(0, I)$) approximates a target data distribution μ_X (the distribution of training images, text sequences, molecules, or whatever the domain requires). Different generative model families correspond to different choices of morphism class, different choices of divergence between $g_*\mu_Z$ and μ_X , and different training procedures.

Before introducing the three main families, we make the categorical setting precise. Consider the Markov category **BorelStoch** of standard Borel spaces and Markov kernels (Chapter 6). A generative model is a morphism $g : Z \rightarrow \mathcal{G}(X)$ in **BorelStoch**, where $\mathcal{G}(X)$ denotes the Giry monad applied to X . A *deterministic* generative model is the special case where $g : Z \rightarrow X$ is a measurable function (a morphism in **Meas**), viewed as the deterministic kernel $g_{\text{det}}(A \mid z) = \mathbf{1}_{g(z) \in A}$. In the Para construction **Para(BorelStoch)**, a *parametric* generative model is a triple (P, Z, g) where P is the parameter space, Z is the latent space, and $g : P \times Z \rightarrow \mathcal{G}(X)$ is the implementation function. Training corresponds to finding the $p \in P$ that makes $g(p, \cdot)_*\mu_Z$ closest to μ_X in a chosen divergence.

The *category of generative models* **GenMod** has:

- Objects: pairs (Z, μ_Z) of latent spaces and reference distributions.
- Morphisms $(Z, \mu_Z) \rightarrow (X, \mu_X)$: parametric families $g_\theta : Z \rightarrow X$ with $\theta \in \Theta$ and a divergence value $D(g_{\theta*}\mu_Z \parallel \mu_X) \geq 0$.
- Composition: $(Z, \mu_Z) \rightarrow (Y, \mu_Y) \rightarrow (X, \mu_X)$ composes the latent codes, useful for hierarchical models where the latent of one model feeds another.

Morphisms in **GenMod** carry a preorder (“ g is at least as good a generative model as h if $D(g_*\mu_Z\|\mu_X) \leq D(h_*\mu_Z\|\mu_X)$ ”), making **GenMod** a category enriched over the poset $([0, \infty], \geq)$ — a Lawvere metric category (Chapter 7). Training a generative model is an optimisation problem: find the morphism of minimum divergence in the enriched category.

23.2 GANs as Morphisms in the Category of Open Games

A *Generative Adversarial Network* (GAN; Goodfellow et al., 2014) introduces a second network — the discriminator — to define the training objective implicitly. Rather than minimising an explicit divergence $D(g_{\theta*}\mu_Z\|\mu_X)$, GAN training sets up a two-player game in which the generator and discriminator have opposing objectives.

Formally, the GAN consists of:

- A *generator* $G_{\theta} : Z \rightarrow X$, a morphism in **Para(Smooth)** with parameter $\theta \in \Theta$.
- A *discriminator* $D_{\phi} : X \rightarrow [0, 1]$, another morphism in **Para(Smooth)** with parameter $\phi \in \Phi$.

The GAN training objective is the minimax game:

$$\min_{\theta} \max_{\phi} V(\theta, \phi) = \mathbb{E}_{x \sim \mu_X} [\log D_{\phi}(x)] + \mathbb{E}_{z \sim \mu_Z} [\log(1 - D_{\phi}(G_{\theta}(z)))].$$

This is a morphism $V : \Theta \times \Phi \rightarrow \mathbb{R}$ in **Smooth**, and the training objective is to find the Nash equilibrium (θ^*, ϕ^*) of the two-player zero-sum game defined by V .

This is precisely an *open game* in the sense of Chapter 10. Let us make the correspondence explicit. The open game $\mathcal{G}_{\text{GAN}} : (X \times Z, \Theta \times \Phi) \rightarrow (\mathbb{R}, \mathbb{R})$ has:

- *Contexts*: the joint distribution $\mu_X \times \mu_Z$ over real and latent samples.
- *Play function*: $P((\theta, \phi), (x, z)) = D_{\phi}(G_{\theta}(z)) \in [0, 1]$, computing the discriminator’s estimate.
- *Equilibrium*: $E((x, z), V) = \{(\theta^*, \phi^*) : V(\theta^*, \phi^*) \text{ is a Nash equilibrium}\}$.

The composition of the generator game $\mathcal{G}_G$ and the discriminator game $\mathcal{G}_D$ in the category of open games gives the full GAN game $\mathcal{G}_{\text{GAN}} = \mathcal{G}_D \circ \mathcal{G}_G$.

Theorem 23.1 (Goodfellow et al., 2014; categorical form). *At the Nash equilibrium (θ^*, ϕ^*) of the GAN game, the following hold:*

1. *The optimal discriminator (given a fixed generator) is $D_{\phi^*}^*(x) = \frac{\mu_X(x)}{\mu_X(x) + G_{\theta^*}\mu_Z(x)}$, the ratio of the data density to the sum of data and generated densities.*
2. *The optimal generator (given the optimal discriminator) minimises the Jensen-Shannon divergence: $G_{\theta^*}\mu_Z = \mu_X$, so $D_{\phi^*}^*(x) = \frac{1}{2}$ everywhere.*
3. *The global Nash equilibrium is $G_{\theta^*}\mu_Z = \mu_X$ with $V(\theta^*, \phi^*) = -\log 4$.*

Proof sketch. Fix θ and optimise over ϕ . The optimal discriminator maximises $V(\theta, \phi)$ pointwise: $D_{\phi}^*(x) = \arg \max_{d \in [0, 1]} \mu_X(x) \log d + G_{\theta}\mu_Z(x) \log(1 - d) = \frac{\mu_X(x)}{\mu_X(x) + G_{\theta}\mu_Z(x)}$,

giving the ratio formula. Substituting into V , the objective for the generator becomes $-\log 4 + 2 \cdot \text{JSD}(\mu_X \| G_{\theta*} \mu_Z)$, where JSD is the Jensen-Shannon divergence. Since $\text{JSD} \geq 0$ with equality iff the distributions coincide, the minimum over θ is achieved when $G_{\theta*} \mu_Z = \mu_X$. $\square$

The game-theoretic instability of GAN training — mode collapse, oscillation, and non-convergence — has a precise categorical interpretation. The Nash equilibrium of the min-max game is not generally a global minimiser of any single objective: it is a saddle point of V . Gradient descent-ascent dynamics (alternately updating θ by gradient descent on V and ϕ by gradient ascent) can cycle around the Nash equilibrium without converging, unless the game satisfies a *monotonicity* condition:

Definition 23.2 (Monotone Game). A two-player game with objective $V : \Theta \times \Phi \rightarrow \mathbb{R}$ is *monotone* if the joint gradient operator $F(\theta, \phi) = (\nabla_{\theta} V(\theta, \phi), -\nabla_{\phi} V(\theta, \phi))$ is a monotone operator: $\langle F(\theta_1, \phi_1) - F(\theta_2, \phi_2), (\theta_1 - \theta_2, \phi_1 - \phi_2) \rangle \geq 0$ for all $(\theta_1, \phi_1), (\theta_2, \phi_2)$.

Monotonicity of the GAN game implies convergence of gradient descent-ascent to the Nash equilibrium (Facchinei-Pang, 2003). The categorical structure of the open game (Chapter 10) identifies monotonicity as the condition that makes the game’s equilibrium functor a right adjoint: the game has a unique Nash equilibrium that is the terminal object in the category of strategy pairs, and gradient dynamics converge to it.

The Wasserstein GAN and Enriched Morphisms

The original GAN uses the Jensen-Shannon divergence implicitly. The *Wasserstein GAN* (WGAN; Arjovsky et al., 2017) replaces it with the Wasserstein-1 distance, which has better gradient properties (non-zero gradients everywhere, even when the supports of μ_X and $G_{\theta*} \mu_Z$ are disjoint).

The Wasserstein-1 distance is defined by Kantorovich-Rubinstein duality:

$$W_1(\mu_X, G_{\theta*} \mu_Z) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mu_X}[f(x)] - \mathbb{E}_{z \sim \mu_Z}[f(G_{\theta}(z))],$$

where the supremum is over all 1-Lipschitz functions $f : X \rightarrow \mathbb{R}$. The categorical significance is profound: W_1 is precisely the *enriched hom-object* of the Lawvere metric category of probability distributions (Chapter 7). Recall that a Lawvere metric space is a category enriched over $([0, \infty], \geq, +)$, and the hom-object $d(\mu, \nu) \in [0, \infty]$ is the “distance” between objects. For the category of probability measures on a metric space (X, d) , the Wasserstein distance $W_1(\mu, \nu)$ is the hom-object in the Lawvere metric sense: it is the largest metric on $\mathcal{G}(X)$ that is compatible with the metric on X via the Lipschitz condition.

The WGAN training objective is:

$$\min_{\theta} \max_{\phi: 1\text{-Lip}} \mathbb{E}_{x \sim \mu_X}[D_{\phi}(x)] - \mathbb{E}_{z \sim \mu_Z}[D_{\phi}(G_{\theta}(z))].$$

Here the discriminator D_{ϕ} is constrained to be a 1-Lipschitz function — a morphism in the enriched category of metric spaces with Lipschitz constant at most 1. Enforcing this con-

straint (via gradient penalty, spectral normalisation, or clipping) ensures that the WGAN training objective approximates $W_1(\mu_X, G_{\theta*}\mu_Z)$, which has continuous and bounded gradients with respect to θ even when the distributions have disjoint support.

The WGAN discriminator is a morphism in the enriched subcategory of 1-Lipschitz functions on X , viewed as a Lawvere metric space. The training objective is the computation of the enriched hom $W_1(\mu_X, G_{\theta*}\mu_Z)$ via its dual representation, and the generator update minimises this hom. The entire WGAN training procedure is a morphism in **Para(Lip₁)**, the Para construction over the enriched category of 1-Lipschitz functions.

23.3 Normalising Flows as Morphisms in **Para(Diff)**

A *normalising flow* is a sequence of invertible smooth transformations:

$$f = f_K \circ f_{K-1} \circ \cdots \circ f_1 : Z \xrightarrow{f_1} Z_1 \xrightarrow{f_2} \cdots \xrightarrow{f_K} X,$$

mapping the reference distribution μ_Z to the model distribution $f_*\mu_Z$ via a diffeomorphism $f = f_K \circ \cdots \circ f_1$. The change-of-variables formula gives the model's log-density:

$$\log p_f(x) = \log p_Z(f^{-1}(x)) + \sum_{k=1}^K \log |\det J_{f_k^{-1}}(f_k \circ \cdots \circ f_1(f^{-1}(x)))|,$$

where $J_{f_k^{-1}}$ is the Jacobian of the k -th inverse map. Maximum likelihood training maximises $\mathbb{E}_{x \sim \mu_X} [\log p_f(x)]$ over the parameters of each f_k .

The categorical structure is the following. Let **Diff** be the category of smooth manifolds and diffeomorphisms. The composition of diffeomorphisms is again a diffeomorphism (group property of **Diff**), and the Jacobian of the composition is the product of Jacobians:

$$J_{f_K \circ \cdots \circ f_1}(z) = J_{f_K}(f_{K-1} \circ \cdots \circ f_1(z)) \cdots J_{f_2}(f_1(z)) J_{f_1}(z).$$

The determinant factorises: $|\det J_f(z)| = \prod_k |\det J_{f_k}(z_k)|$ (where $z_k = f_k \circ \cdots \circ f_1(z)$). This is the chain rule for Jacobians, i.e., the functoriality of the tangent functor $T : \mathbf{Diff} \rightarrow \mathbf{Diff}$ (Chapter 1).

A normalising flow is thus a morphism in **Para(Diff)**: the parameter space is $\Theta = \Theta_1 \times \cdots \times \Theta_K$ (one parameter set per layer), and the implementation is $f_\theta = f_{\theta_K} \circ \cdots \circ f_{\theta_1} : Z \rightarrow X$. The backpropagation functor $\mathbf{BP} : \mathbf{Para(Diff)} \rightarrow \mathbf{Lens(Diff)}$ sends f_θ to the lens:

$$\begin{aligned} \mathbf{get}(\theta, z) &= (f_\theta(z), \log |\det J_{f_\theta}(z)|) \quad (\text{forward pass: value and log-Jacobian}), \\ \mathbf{put}((\theta, z), \delta) &= \left(\frac{\partial \log |\det J_{f_\theta}|}{\partial \theta} \Big|_z \delta, J_{f_\theta}(z)^{-\top} \delta \right) \quad (\text{backward pass}). \end{aligned}$$

The residual state saved during the forward pass includes the intermediate activations $z_1, \dots, z_{K-1}$ and the Jacobians $J_{f_k}(z_{k-1})$ for each layer, because the backward pass needs them to compute the gradient.

Example 23.3 (Coupling Layers and Triangular Jacobians). The RealNVP architecture

(Dinh et al., 2017) uses *affine coupling layers*: split the input $z = (z_A, z_B)$ into two halves, compute scaling and shifting from one half, and apply them to the other:

$$f_\theta(z_A, z_B) = (z_A, z_B \odot \exp(s_\theta(z_A)) + t_\theta(z_A)),$$

where $s_\theta, t_\theta : \mathbb{R}^{d/2} \rightarrow \mathbb{R}^{d/2}$ are arbitrary neural networks. The Jacobian of this transformation is lower-triangular with the $\exp(s_\theta(z_A))$ values on the diagonal (since $\partial f_B / \partial z_A = \partial_B s_\theta \cdot z_B + \partial_B t_\theta$ contributes off-diagonal terms, while $\partial f_B / \partial z_B = \text{diag}(\exp(s_\theta(z_A)))$). The log-determinant is then $\sum_i s_\theta(z_A)_i$ — computable in $O(d)$ time, not $O(d^3)$.

The categorical content: coupling layers are morphisms in the subcategory $\mathbf{Diff}_\Delta \subset \mathbf{Diff}$ of diffeomorphisms with triangular Jacobian. The composition of two coupling layers (with alternating splits) is still in $\mathbf{Diff}_\Delta$ (by the product formula for triangular determinants), so the entire flow is a morphism in $\mathbf{Para}(\mathbf{Diff}_\Delta)$.

Example 23.4 (Invertible 1×1 Convolutions). The Glow architecture (Kingma and Dhariwal, 2018) augments coupling layers with learned invertible 1×1 convolutions $W \in GL(C)$ applied channel-wise: $f_W(z)_{c,h,w} = \sum_{c'} W_{cc'} z_{c',h,w}$. The log-Jacobian is $HW \log |\det W|$ (for an $H \times W$ spatial grid), computable via the LU decomposition of $W = PLU$ (with P a fixed permutation matrix). The categorical structure: $GL(C)$ acts on the feature space by the group representation $\rho(W) = W \otimes I_{HW}$ (Kronecker product with the identity on the spatial dimensions), and the 1×1 convolution is a morphism in $\mathbf{Para}(\mathbf{Diff})$ parametrised by $W \in GL(C)$.

23.4 Denoising Diffusion as a Markov Chain Morphism

Denoising Diffusion Probabilistic Models (DDPMs; Ho et al., 2020) define a two-phase process. The *forward process* gradually destroys the data structure by adding Gaussian noise in T small steps; the *reverse process* learns to reconstruct the data by iteratively denoising.

The Forward Process as a Composite Markov Kernel

The forward process is a Markov chain:

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^T q(x_t | x_{t-1}), \quad q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I),$$

with a fixed variance schedule $0 < \beta_1 < \dots < \beta_T < 1$. In the Markov category **BorelStoch** (Chapter 6), each $q(x_t | x_{t-1})$ is a Markov kernel $\mathbf{K}_t : X \rightarrow \mathcal{G}(X)$. The full forward process is the composite Markov kernel:

$$q_{0:T} = \mathbf{K}_T \circ \mathbf{K}_{T-1} \circ \dots \circ \mathbf{K}_1 : X \rightarrow \mathcal{G}(X^T).$$

By the Markov property and the Gaussian convolution formula, the marginal at step t is:

$$q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)I), \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s).$$

As $t \rightarrow T$ and $\bar{\alpha}_T \rightarrow 0$, the forward process erases all information about x_0 , leaving pure Gaussian noise $q(x_T | x_0) \approx \mathcal{N}(0, I)$.

The Reverse Process as a Parametric Markov Kernel

The reverse process $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t)$ is a morphism in **Para(BorelStoch)**: a parametrised Markov kernel $X_T \rightarrow \mathcal{G}(X_0)$, with each step $p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$ learned to approximate the true reverse conditional.

The true reverse conditional (which is tractable when x_0 is known) is:

$$q(x_{t-1} | x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I),$$

with $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$ and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. The network $\varepsilon_\theta(x_t, t)$ learns to predict the noise ε such that $x_0 = (x_t - \sqrt{1 - \bar{\alpha}_t}\varepsilon)/\sqrt{\bar{\alpha}_t}$, and the training objective is:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}_{t, x_0, \varepsilon} [\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon, t)\|^2].$$

The categorical structure: $\mathcal{L}_{\text{DDPM}}$ is a morphism in **Para(BorelStoch)** (a parametrised expected squared error), and minimising it is the problem of finding the morphism ε_θ that best approximates the score function (the gradient of the log-density of $q(x_t)$).

The Score Function as a Natural Transformation

The *score function* of a distribution p on a manifold M is the gradient of its log-density: $\nabla_x \log p(x) \in T_x M$. As a function of p , the score function is a *natural transformation*:

$$s : \mathcal{G}(M) \Rightarrow \mathfrak{X}(M), \quad s_M(p)(x) = \nabla_x \log p(x),$$

from the functor $\mathcal{G}(M)$ (probability distributions on M) to the functor $\mathfrak{X}(M)$ (smooth vector fields on M). Naturality means: for any diffeomorphism $\phi : M \rightarrow N$, $s_N(\phi_* p) = (\phi_*)_* s_M(p)$ (the score of the pushforward distribution is the pushforward of the original score vector field). This is the basis-independence of the score function.

The DDPM denoising network $\varepsilon_\theta(x_t, t)$ is a learnable approximation to the score function $s_{q_{x_t}}(x_t) = \nabla_{x_t} \log q(x_t)$, where $q(x_t) = \int q(x_t | x_0) q(x_0) dx_0$ is the marginal at time t . Score matching (Hyvärinen, 2005) provides the training objective:

$$J_{\text{SM}}(\theta) = \mathbb{E}_{x_t \sim q} [\|s_\theta(x_t, t) - \nabla_{x_t} \log q(x_t | x_0)\|^2] + \text{const},$$

which avoids computing the intractable marginal score by conditioning on x_0 (using the tractable conditional score $\nabla_{x_t} \log q(x_t | x_0) = -(x_t - \sqrt{\bar{\alpha}_t}x_0)/((1 - \bar{\alpha}_t))$). The DDPM training objective $\mathcal{L}_{\text{DDPM}}$ is equivalent to J_{SM} up to a time-dependent reweighting.

23.5 Continuous-Time Diffusion and the SDE Category

The DDPM can be extended to continuous time (Song et al., 2021), giving a unifying framework based on stochastic differential equations. The forward process is the SDE:

$$dx_t = f(x_t, t) dt + g(t) dW_t,$$

where $f(x_t, t) = -\frac{1}{2}\beta(t)x_t$ (drift) and $g(t) = \sqrt{\beta(t)}$ (diffusion coefficient) for the DDPM variance-exploding schedule. Here W_t is a standard Wiener process (Brownian motion).

The SDE defines a Markov process $\{x_t\}_{t \in [0, T]}$, and the associated *probability flow ODE*:

$$\frac{dx}{dt} = f(x, t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)$$

has the same marginal distributions p_t as the SDE but is deterministic: it is an ODE (a morphism in **ODE**, Chapter 24) whose solution at time T is the noise and whose solution at time 0 is the data. The probability flow ODE can be solved *backwards* (from noise to data) using any ODE solver, giving the DDIM (Song et al., 2021) sampler.

The categorical structure of the continuous-time diffusion is the following. Define the category **SDE** whose objects are smooth manifolds and whose morphisms are Itô SDEs (pairs of drift $f : M \times [0, T] \rightarrow TM$ and diffusion $g : M \times [0, T] \rightarrow \mathbf{Hom}(\mathbb{R}^d, TM)$ satisfying appropriate growth conditions). The SDE is a morphism $\text{SDE}_{f,g} : (M, x_0) \rightarrow (M, x_T)$ in **SDE**, sending the initial condition x_0 to the terminal distribution p_T via the Itô map. The forward diffusion is a morphism in **SDE**; the learned reverse process is a morphism in **Para(SDE)**.

Anderson’s reverse-time SDE (Anderson, 1982) states that the time-reversal of the forward SDE is:

$$d\bar{x}_t = [f(\bar{x}_t, T - t) - g(T - t)^2 \nabla_{\bar{x}_t} \log p_{T-t}(\bar{x}_t)] dt + g(T - t) d\bar{W}_t,$$

which is again a morphism in **SDE** (running from $t = 0$ to $t = T$ in reverse time). The learned DDPM reverse process approximates this morphism by replacing $\nabla \log p_{T-t}$ with the learned score network $s_\theta(\bar{x}_t, T - t)$.

Theorem 23.5 (Score Matching and the Backpropagation Lens). *The score matching objective $J_{\text{SM}}(\theta)$ is the squared Frobenius norm of the difference between two morphisms in **Para(SDE)**: the true score $s_{q_t} : \mathcal{G}(M) \rightarrow \mathfrak{X}(M)$ and the learned score $s_\theta : M \times [0, T] \rightarrow TM$. The gradient of J_{SM} with respect to θ is computed by the backpropagation functor $\text{BP} : \mathbf{Para}(\mathbf{SDE}) \rightarrow \mathbf{Lens}(\mathbf{SDE})$, whose lens component gives the gradient of the loss with respect to the network parameters.*

23.6 The Unified View via Categorical Optimal Transport

Normalising flows, GANs, and DDPMs are all methods for transporting the reference measure μ_Z to the data measure μ_X . Optimal transport (Monge, 1781; Kantorovich,

1942) provides the unifying categorical framework: it seeks the most efficient morphism for this transport, where “efficiency” is measured by a cost function $c : X \times Z \rightarrow [0, \infty)$.

The *Monge problem* is to find the deterministic transport map $T : Z \rightarrow X$ (a morphism in **Smooth** or **Meas**) minimising:

$$\text{OT}_c(\mu_Z, \mu_X) = \inf_{T: T_*\mu_Z = \mu_X} \int_Z c(T(z), z) d\mu_Z(z).$$

The *Kantorovich relaxation* replaces the deterministic map with a transport plan $\gamma \in \mathcal{G}(Z \times X)$ with marginals $\pi_{Z*}\gamma = \mu_Z$ and $\pi_{X*}\gamma = \mu_X$, minimising $\int_{Z \times X} c(z, x) d\gamma(z, x)$.

The Wasserstein- p distance $W_p(\mu_Z, \mu_X) = \text{OT}_{d^p}(\mu_Z, \mu_X)^{1/p}$ (with cost $c = d^p$) is the value of the Kantorovich problem. It is the hom-object in the category of probability measures on (X, d) enriched over $([0, \infty], \geq, +)$ — a Lawvere metric (Chapter 7). The Kantorovich duality theorem is the statement that $W_1(\mu_Z, \mu_X) = \sup_{\|f\|_L \leq 1} [\mathbb{E}_{\mu_X}[f] - \mathbb{E}_{\mu_Z}[f]]$, which is the dual representation of the enriched hom as a supremum in the dual enriched category.

The three generative model families correspond to three approximation strategies for the optimal transport map:

- *Flows*: approximate the Monge map $T^* : Z \rightarrow X$ directly as a composition of diffeomorphisms. The approximation quality is limited by the family of diffeomorphisms available.
- *GANs*: approximate the Monge map indirectly, by training against a discriminator that estimates the optimal transport cost. The WGAN uses the Kantorovich dual representation.
- *DDPMs*: approximate the Monge map via a path of small steps (the stochastic interpolant), each of which transports the measure a small amount. The diffusion path is not the optimal transport path (which would be a straight line, the McCann interpolant), but it is computable by local denoising operations.

Proposition 23.6 (Flow Matching as Optimal Transport). *The Flow Matching framework (Lipman et al., 2022; Albergo and Vanden-Eijnden, 2022) unifies flows and diffusion by training a velocity field $v_\theta : X \times [0, 1] \rightarrow TX$ (a time-dependent vector field) to match the conditional flow $u_t(x | x_1) = (x_1 - x)/|x_1 - x|$ (the straight-line path from noise $x_0 \sim \mu_Z$ to data $x_1 \sim \mu_X$). The optimal velocity field (minimising the MSE loss) is the Riemannian-gradient flow of the W_2 Wasserstein distance, making Flow Matching the continuous-time approximation to the optimal Monge transport.*

Categorically, Flow Matching is a morphism in **Para(ODE)** (not **SDE**, since the optimal transport path is deterministic): the velocity field v_θ is the parametrised vector field of a Neural ODE (Chapter 24), and its training objective is the squared distance in the enriched category of vector fields. The resulting generative model is the flow of the trained ODE, which maps noise to data along paths that approximate the Monge transport.

Chapter 24

Residual Architectures, Skip Connections, and Categorical Constructions

24.1 The Residual Connection as a Bimonoid Morphism

The fundamental operation of a ResNet block (He et al., 2016) is $y = F(x) + x$, where $F : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a neural network sub-block and $x \in \mathbb{R}^d$ is the input. This appears simple, but its categorical structure is rich and explains both the practical success and the theoretical properties of residual architectures.

In $\mathbf{FVect}_{\mathbb{R}}$ with the direct sum monoidal structure $(\oplus, 0)$, the residual connection factors as:

$$\mathbb{R}^d \xrightarrow{\Delta} \mathbb{R}^d \oplus \mathbb{R}^d \xrightarrow{F \oplus \text{id}_{\mathbb{R}^d}} \mathbb{R}^d \oplus \mathbb{R}^d \xrightarrow{\nabla} \mathbb{R}^d, \quad (24.1)$$

where $\Delta(x) = (x, x)$ is the diagonal (copy morphism) and $\nabla(u, v) = u + v$ is the codiagonal (addition morphism). The vector space $\mathbb{R}^d$ is a *bimonoid* in $(\mathbf{FVect}_{\mathbb{R}}, \oplus, 0)$: it is simultaneously a monoid $(\mathbb{R}^d, \nabla, 0)$ (with multiplication given by addition and unit the zero vector) and a comonoid $(\mathbb{R}^d, \Delta, \varepsilon)$ (with comultiplication given by the diagonal and counit the zero map $\varepsilon : \mathbb{R}^d \rightarrow 0$). The Frobenius law $(\nabla \oplus \text{id}) \circ (\text{id} \oplus \Delta) = \nabla \circ \Delta = (\text{id} \oplus \nabla) \circ (\Delta \oplus \text{id})$ holds automatically in $\mathbf{FVect}_{\mathbb{R}}$.

The factorisation (24.1) shows that the residual connection $\text{Res}(F) = \nabla \circ (F \oplus \text{id}) \circ \Delta$ is a morphism in the bimonoid structure of $\mathbb{R}^d$.

Proposition 24.1 (Properties of the Residual Map). *Let $\mathcal{C}$ be a symmetric monoidal category in which every object A is a bimonoid with copy $\Delta_A : A \rightarrow A \otimes A$ and add $\nabla_A : A \otimes A \rightarrow A$. Then the residual map $\text{Res} : \text{End}(A) \rightarrow \text{End}(A)$, $F \mapsto \nabla_A \circ (F \otimes \text{id}_A) \circ \Delta_A$, satisfies:*

1. $\text{Res}(0_A) = \text{id}_A$: the zero morphism becomes the identity.
2. $\text{Res}(\text{id}_A) = 2\text{id}_A$: the identity is doubled.
3. $\text{Res}(F + G) = \text{Res}(F) + \text{Res}(G) - \text{id}_A$ (for additive $\mathcal{C}$).

4. *Res commutes with conjugation:* $\text{Res}(\phi \circ F \circ \phi^{-1}) = \phi \circ \text{Res}(F) \circ \phi^{-1}$ for invertible ϕ .

Proof. For (1): $\nabla \circ (0 \oplus \text{id}) \circ \Delta(x) = \nabla(0, x) = x = \text{id}_A(x)$. For (2): $\nabla \circ (\text{id} \oplus \text{id}) \circ \Delta(x) = \nabla(x, x) = 2x$. For (3): $(F+G) \oplus \text{id} = (F \oplus \text{id}) + (G \oplus \text{id}) - (0 \oplus \text{id})$, and applying $\nabla \circ (-) \circ \Delta$ distributes over the sum by bilinearity of ∇ and Δ , giving $\text{Res}(F) + \text{Res}(G) - \text{Res}(0) = \text{Res}(F) + \text{Res}(G) - \text{id}_A$. For (4): direct calculation using the naturality of Δ and ∇ with respect to invertible morphisms. $\square$

Property (1) is the categorical explanation of why residual connections mitigate the vanishing gradient problem. If a block F fails to learn any useful transformation ($F \approx 0$), the output of $\text{Res}(F)$ is approximately id_A — the identity function. The gradient of id_A with respect to its input is the identity matrix (all eigenvalues equal to 1), so gradients pass through unchanged. Without the residual connection, a block $F \approx 0$ would have near-zero gradients at all preceding layers, making deep network training impossible. The residual connection guarantees a “gradient highway” through the network.

24.2 The Backward Pass through a Residual Connection

The backpropagation functor $\text{BP} : \mathbf{Para}(\mathbf{Smooth}) \rightarrow \mathbf{Lens}(\mathbf{Smooth})$ applied to $\text{Res}(F_\theta) = F_\theta + \text{id}$ gives a lens whose forward and backward passes are:

$$\begin{aligned} \text{get}(\theta, x) &= F_\theta(x) + x, \\ \text{put}((\theta, x), \delta y) &= \left(\frac{\partial F_\theta}{\partial \theta}(x)^\top \delta y, \frac{\partial F_\theta}{\partial x}(x)^\top \delta y + \delta y \right). \end{aligned}$$

The backward component has two terms: $(\partial F_\theta / \partial x)^\top \delta y$ (the gradient through the nonlinear branch F_θ) and δy (the gradient through the skip connection). The skip connection thus adds δy directly to the gradient, guaranteeing that the gradient reaching the input of the block is at least $\|\delta y\|$ (the norm of the upstream gradient), regardless of the properties of F_θ .

This is the categorical dualisation of Property (1) in Proposition 24.1: if $F_\theta \approx 0$, the gradient through F_θ is approximately zero, but the gradient through the skip connection δy survives. The entire ResNet backward pass is thus the sequential composition of L lenses, each of which passes the gradient through both a nonlinear branch and a direct skip, and the composition formula (Chapter 3) guarantees that the backward pass correctly aggregates all contributions.

24.3 DenseNet, Pre-Activation, and Generalised Skip Connections

The *DenseNet* (Huang et al., 2017) generalises the single skip connection of the ResNet by concatenating *all* previous layer outputs:

$$x_\ell = H_\ell([x_0, x_1, \dots, x_{\ell-1}]),$$

where $[x_0, \dots, x_{\ell-1}] \in \mathbb{R}^{(\ell-1) \cdot d}$ is the concatenation of all previous feature maps and $H_\ell : \mathbb{R}^{(\ell-1)d} \rightarrow \mathbb{R}^d$ is the ℓ -th layer.

Categorically, the DenseNet uses the *coproduct* $\bigoplus_{k < \ell} \mathbb{R}_k^d$ in $\mathbf{FVect}_{\mathbb{R}}$ (the direct sum of all previous feature spaces), rather than the bimonoid codiagonal. The layer H_ℓ is a morphism $H_\ell : \bigoplus_{k < \ell} \mathbb{R}_k^d \rightarrow \mathbb{R}_\ell^d$ in $\mathbf{Para}(\mathbf{FVect}_{\mathbb{R}})$, and the canonical injection $\iota_k : \mathbb{R}_k^d \hookrightarrow \bigoplus_{j < \ell} \mathbb{R}_j^d$ is the coproduct inclusion. The DenseNet forward pass is the colimit of the diagram of all layer outputs in $\mathbf{FVect}_{\mathbb{R}}$, and the backward pass is the limit of the corresponding diagram in the cotangent category.

The *pre-activation ResNet* (He et al., 2016, second paper) reorders the operations within each block to apply batch normalisation and activation before the weight layers: $y = x + W_2 \sigma(\text{BN}(W_1 \sigma(\text{BN}(x))))$. The categorical significance is that the pre-activation ordering makes the block $F_\theta = W_2 \sigma(\text{BN}(W_1 \sigma(\text{BN}(\cdot))))$ a morphism in the category of smooth maps *before* the residual addition, so the identity path id_A in the residual factorisation is truly “clean” (unnormalised and unactivated). This improves gradient flow and enables training of extremely deep networks (1001 layers were demonstrated).

24.4 Neural ODEs as the Continuous Limit

The deep connection between residual networks and ordinary differential equations (E, 2017; Chen et al., 2018) is one of the most elegant results in the categorical theory of deep learning. A ResNet with L blocks of step size $h = T/L$ can be written as:

$$x_{\ell+1} = x_\ell + h \cdot F_\theta(x_\ell, \ell h) \approx x((\ell+1)h),$$

where $x(t)$ satisfies the ODE $\dot{x}(t) = F_\theta(x(t), t)$ with $x(0) = x_0$. As $L \rightarrow \infty$ and $h \rightarrow 0$, the ResNet converges to the *Neural ODE*: the flow $\Phi_{[0,T]}^\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$ of the ODE $\dot{x} = F_\theta(x, t)$ over the interval $[0, T]$.

The Neural ODE is a morphism in $\mathbf{Para}(\mathbf{ODE})$: the parameter space is Θ (the weights of the neural network defining F_θ), and the implementation is the solution map $\Phi_T^\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$ obtained by integrating the ODE from 0 to T . The composition of two Neural ODEs (with independent vector fields on disjoint time intervals $[0, T_1]$ and $[T_1, T_2]$) is again a Neural ODE (on $[0, T_2]$), by the group law for flows: $\Phi_{[T_1, T_2]}^{\theta_2} \circ \Phi_{[0, T_1]}^{\theta_1} = \Phi_{[0, T_2]}^{(\theta_1, \theta_2)}$.

The backpropagation functor applied to a Neural ODE gives the *adjoint method*: rather than storing all intermediate activations (as in a finite-depth ResNet), the adjoint method solves a second ODE (the adjoint ODE) backwards in time to compute the gradient. The

lens structure of the adjoint method is:

$$\begin{aligned} \mathbf{get}(\theta, x_0) &= x(T) = \Phi_T^\theta(x_0), \\ \mathbf{put}((\theta, x_0), \delta x_T) &= \left(\int_T^0 \frac{\partial F_\theta}{\partial \theta}(x(t), t)^\top a(t) dt, a(0) \right), \end{aligned}$$

where the *adjoint state* $a(t)$ satisfies the adjoint ODE:

$$\frac{da}{dt} = -\frac{\partial F_\theta}{\partial x}(x(t), t)^\top a(t), \quad a(T) = \delta x_T.$$

The adjoint ODE is a morphism in **ODE** on the cotangent bundle $T^*\mathbb{R}^d$, and the integral $\int_T^0 (\partial F_\theta / \partial \theta)^\top a(t) dt$ is the gradient with respect to θ . The correctness of the adjoint method (i.e., that it computes the true gradient) follows from the chain rule in the CRDC framework (Chapter 3) applied to the ODE flow.

24.5 Normalisation Layers and the Geometry of Activation Distributions

Batch Normalisation (Ioffe and Szegedy, 2015) and its successors (Layer Norm, Group Norm, Instance Norm, RMS Norm) are normalisation operations that standardise the distribution of activations within each layer, improving training stability and speed. We give a unified categorical account.

Let $\mu : \mathbb{R}^{B \times d} \rightarrow \mathbb{R}^d$ and $\sigma : \mathbb{R}^{B \times d} \rightarrow \mathbb{R}_{>0}^d$ be the batch mean and standard deviation (for Batch Norm), or the layer mean and standard deviation (for Layer Norm). The normalisation operation is:

$$\text{Norm}(X) = \frac{X - \mathbf{1}\mu(X)^\top}{\mathbf{1}\sigma(X)^\top} \odot \Gamma + \mathbf{1}B^\top,$$

where $\Gamma, B \in \mathbb{R}^d$ are learned scale and shift parameters (in matrix form), $\mathbf{1}$ is the all-ones vector, and $\odot$ is element-wise multiplication.

The categorical account uses the concept of a *group quotient*. The normalisation defines an equivalence relation on the activation space: $X \sim Y$ if and only if there exist $\lambda \in \mathbb{R}_{>0}^d$ and $\mu \in \mathbb{R}^d$ such that $X = Y \odot \lambda + \mathbf{1}\mu^\top$ (element-wise rescaling and shifting in the feature dimension). The quotient $\mathbb{R}^{B \times d} / \sim$ is the space of *equivalence classes of activation matrices under feature-wise affine transformation*, and the normalisation $\text{Norm} : \mathbb{R}^{B \times d} \rightarrow \mathbb{R}^{B \times d} / \sim$ is the quotient map.

More precisely, the group $G = \mathbb{R}_{>0}^d \times \mathbb{R}^d$ (positive rescaling and shifting, one scale-shift pair per feature) acts on $\mathbb{R}^{B \times d}$ by $((\lambda, \mu), X) \mapsto X \odot \lambda + \mathbf{1}\mu^\top$. The normalisation is the G -equivariant map that selects the canonical representative of each G -orbit: the one with mean 0 and standard deviation 1. The learned parameters (Γ, B) are the learned affine transformation applied after normalisation, mapping the canonical representative to the optimal point within the orbit.

Proposition 24.2 (Normalisation as Categorical Quotient). *Layer Normalisation is the quotient map $\pi : \mathbb{R}^d \setminus \{0\}^d \rightarrow (\mathbb{R}^d \setminus \{0\}^d)/G$ for the group $G = \mathbb{R}_{>0} \times \mathbb{R}$ acting by $(\lambda, \mu) \cdot x = \lambda x + \mu \mathbf{1}$. It factors as:*

$$\mathbb{R}^d \xrightarrow{x \mapsto x - \bar{x} \mathbf{1}} H \xrightarrow{v \mapsto v / \|v\|} S^{d-2} \cap H \xrightarrow{(\Gamma, B)} \mathbb{R}^d,$$

where $H = \{v \in \mathbb{R}^d : \sum_i v_i = 0\}$ is the zero-mean hyperplane (a linear projection, hence a morphism in $\mathbf{FVect}_{\mathbb{R}}$), $S^{d-2} \cap H$ is the unit sphere in H (a smooth morphism in $\mathbf{Smooth}$), and the final step is the learned affine reparametrisation (a morphism in $\mathbf{Para}(\mathbf{FVect}_{\mathbb{R}})$).

The invariance $\text{LN}(\lambda x + \mu \mathbf{1}) = \text{LN}(x)$ is the statement that LN factors through the quotient map π : it is constant on G -orbits. The gradient of LN with respect to x is the orthogonal projection onto the tangent space of $S^{d-2} \cap H$ at the normalised point, which is a morphism in $\mathbf{Smooth}$ and can be computed efficiently.

Chapter 25

Categorical Causal Inference and the Do-Calculus

25.1 Structural Causal Models in the Markov Category

The fundamental challenge of causal inference is to distinguish between *association* (observational relationship) and *causation* (interventional relationship). Observing that X and Y are correlated does not imply that X causes Y ; the correlation might be due to a common cause Z (confounding), or to selection bias, or to the causal direction being reversed.

The *structural causal model* (SCM) framework (Pearl, 2000; Peters, Janzing, Schölkopf, 2017) formalises causation through a set of structural equations: $V_i = f_i(\text{Pa}(V_i), U_i)$ for each variable V_i , where $\text{Pa}(V_i)$ are the direct causes (parents in the causal DAG $\mathcal{G}$) and U_i are exogenous noise variables with a joint distribution $\mathcal{P}_U$.

In the Markov category **BorelStoch** (Chapter 6), an SCM is a pair $(\mathcal{G}, f)$ where:

- $\mathcal{G}$ is a directed acyclic graph (DAG) with vertex set $V = \{V_1, \dots, V_n\}$ and edge set $E \subseteq V \times V$.
- $f = (f_1, \dots, f_n)$ is a collection of Markov kernels, one per variable: $f_i : \prod_{j \in \text{Pa}(i)} \mathcal{X}_j \times \mathcal{U}_i \rightarrow \mathcal{G}(\mathcal{X}_i)$, where $\mathcal{X}_j$ and $\mathcal{U}_i$ are the state spaces of V_j and U_i .

The structural equations are composed in topological order of the DAG (vertices with no parents first), giving the composite Markov kernel $f_{\mathcal{G}} : \prod_i \mathcal{U}_i \rightarrow \mathcal{G}(\prod_i \mathcal{X}_i)$. The observational distribution is $p(V_1, \dots, V_n) = (f_{\mathcal{G}})_* \mathcal{P}_U$.

Theorem 25.1 (Causal Markov Condition, string diagram form). *For an SCM $(\mathcal{G}, f, \mathcal{P}_U)$ with independent exogenous noise ($\mathcal{P}_U = \prod_i \mathcal{P}_{U_i}$), the observational distribution satisfies the factorisation:*

$$p(V_1, \dots, V_n) = \prod_{i=1}^n p(V_i \mid \text{Pa}(V_i)).$$

*In the string diagram calculus of **BorelStoch** (Chapter 6), this factorisation is represented by the commutative diagram where the copy morphisms δ_{V_i} applied to the parent variables*

$\text{Pa}(V_i)$ feed into each structural kernel f_i , and no feedback loops are present (reflecting the acyclicity of $\mathcal{G}$).

25.2 The Do-Calculus as a Category of Modified SCMs

The *do-operator* $do(V_i = v)$ models an external intervention that fixes V_i to the value v , breaking all causal arrows *into* V_i (since V_i 's value is now set by the experimenter, not by its usual causes). In the SCM framework, this corresponds to replacing the structural equation $f_i : \prod_j \mathcal{X}_{\text{Pa}(j)} \times \mathcal{U}_i \rightarrow \mathcal{G}(\mathcal{X}_i)$ with the constant kernel $\kappa_v : * \rightarrow \mathcal{G}(\mathcal{X}_i)$, $\kappa_v = \delta_v$ (the Dirac delta at v).

Define the category $\mathbf{SCM}_{\mathcal{G}}$ of SCMs with a fixed DAG $\mathcal{G}$:

- Objects: collections of structural kernels $(f_1, \dots, f_n)$.
- Morphisms: pointwise domination (one SCM produces distributions that dominate another).

The do-operator is a functor $do_{V_i=v} : \mathbf{SCM}_{\mathcal{G}} \rightarrow \mathbf{SCM}_{\mathcal{G}_{\overline{V_i}}}$ (where $\mathcal{G}_{\overline{V_i}}$ is the DAG with all edges into V_i removed): it sends $(f_1, \dots, f_n)$ to $(f_1, \dots, f_{i-1}, \kappa_v, f_{i+1}, \dots, f_n)$.

The *interventional distribution* is $p(V \setminus V_i \mid do(V_i = v)) = (f_{\mathcal{G}}^{do(V_i=v)})_* \mathcal{P}_{U \setminus U_i}$: the observational distribution of the modified SCM, where $\mathcal{P}_{U \setminus U_i}$ omits the exogenous noise U_i (since V_i is now fixed).

Example 25.2 (Backdoor Criterion). Suppose the causal graph is $Z \rightarrow X \rightarrow Y$ with an additional arrow $Z \rightarrow Y$ (so Z is a confounder). The observational distribution is $p(Y \mid X) = \int p(Y \mid X, Z)p(Z \mid X) dZ$, which mixes the causal effect of X on Y with the confounding effect via Z . The interventional distribution (the true causal effect) is $p(Y \mid do(X = x)) = \int p(Y \mid X = x, Z)p(Z) dZ$ (marginalising over Z with respect to its marginal, not its conditional given X). The *backdoor adjustment* formula $p(Y \mid do(X = x)) = \int p(Y \mid X = x, Z = z)p(Z = z) dz$ is a morphism in **BorelStoch**: it replaces the confounded conditional $p(Z \mid X)$ with the marginal $p(Z)$, blocking the backdoor path $X \leftarrow Z \rightarrow Y$.

25.3 Pearl's Do-Calculus as Rewriting Rules in BorelStoch

Pearl's do-calculus consists of three inference rules that together characterise all identifiable causal effects. Each rule states an equality between two morphisms in **BorelStoch** that holds whenever a specific graphical (d-separation) condition is satisfied.

Theorem 25.3 (Do-Calculus, Pearl 1995). *Let $\mathcal{G}$ be the causal DAG, and let X, Y, Z, W be disjoint sets of variables. The following rules hold (where $\mathcal{G}_{\overline{X}}$ denotes $\mathcal{G}$ with all edges into X removed, and $\mathcal{G}_{\underline{X}}$ denotes $\mathcal{G}$ with all edges out of X removed):*

1. **Rule 1 (Insertion/deletion of observations):**

$$p(Y \mid do(X), Z, W) = p(Y \mid do(X), W) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{\mathcal{G}_{\overline{X}}}.$$

2. **Rule 2 (Action/observation exchange):**

$$p(Y \mid \text{do}(X), \text{do}(Z), W) = p(Y \mid \text{do}(X), Z, W) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{\mathcal{G}_{\overline{X}, \underline{Z}}}.$$

3. **Rule 3 (Deletion of actions):**

$$p(Y \mid \text{do}(X), \text{do}(Z), W) = p(Y \mid \text{do}(X), W) \quad \text{if } (Y \perp\!\!\!\perp Z \mid X, W)_{\mathcal{G}_{\overline{X}, \overline{Z(W)}}},$$

where $Z(W)$ is the set of Z -variables not ancestors of any W -variable in $\mathcal{G}_{\overline{X}}$.

In the Markov category **BorelStoch**, each rule is an equality between two morphisms:

- The left-hand side is a morphism obtained by applying the do -functor to Z and then computing a conditional.
- The right-hand side is the simpler morphism (with Z either observed or absent).
- The d -separation condition is the categorical conditional independence condition of Chapter 6 (commutativity of the copy-map diagram in **BorelStoch**) applied to the modified graph.

Theorem 25.4 (Completeness of the Do-Calculus, Shpitser-Pearl 2006; Huang-Valtorta 2006). *An interventional distribution $p(Y \mid \text{do}(X), W)$ is identifiable from the observational distribution $p(V)$ and the DAG $\mathcal{G}$ (i.e., can be computed without knowledge of the structural equations f_i) if and only if it can be reduced to observational distributions by a finite sequence of applications of the three do-calculus rules.*

The completeness theorem is a normalisation theorem for the rewriting system defined by the three rules in **BorelStoch**: every derivable equality is a valid causal identity, and every valid causal identity is derivable. The categorical proof uses the string diagram calculus: a string diagram identity in **BorelStoch** holds if and only if it can be derived by the axioms of Markov categories (Chapter 6) plus the do-calculus rules, making the do-calculus a complete axiomatisation of causal reasoning in Markov categories.

25.4 Counterfactuals, Twin Networks, and the Category of Possible Worlds

Counterfactuals go beyond interventional distributions: they ask “what would Y have been had X been x' , given that we observed $X = x$?” Counterfactual reasoning requires knowledge of the specific exogenous noise value u that led to the observation, which is in general not identifiable from observational data alone.

The *twin network* construction (Balke-Pearl, 1994) handles counterfactuals by running two copies of the SCM on the same exogenous noise U but with different interventions. The twin network is a span in **BorelStoch**:

$$\mathcal{G}(V_{\text{actual}}) \xleftarrow{f_{\mathcal{G}}^{\text{actual}}} \mathcal{G}(U) \xrightarrow{f_{\mathcal{G}}^{\text{cf}}} \mathcal{G}(V_{\text{cf}}),$$

where $f_{\mathcal{G}}^{\text{actual}}$ uses the original structural equations and $f_{\mathcal{G}}^{\text{cf}}$ uses the do-intervened structural equations. Both morphisms use the same exogenous noise distribution $\mathcal{P}_U$, capturing the shared “background state” of the world in both the actual and counterfactual scenarios.

The counterfactual distribution $p(Y_{x'} \mid X = x)$ (the distribution of Y in the counterfactual world where $X = x'$, given that we observed $X = x$) is computed as:

1. **Abduction:** Update $\mathcal{P}_U$ to the posterior $\mathcal{P}_{U|X=x}$ (using the actual observation).
2. **Action:** Apply $do(X = x')$ to get the modified SCM.
3. **Prediction:** Compute the distribution of Y in the modified SCM with the updated noise distribution.

Each step is a morphism in **BorelStoch**: abduction is the Bayesian update (composition with the conditional kernel $p(U \mid X = x)$), action is the do-functor (replacement of the structural kernel), and prediction is the pushforward along the modified structural morphism.

Define the *category of possible worlds* **PossWorlds**($\mathcal{G}$): objects are pairs $(V_{\text{actual}}, do(\mathbf{X} = \mathbf{x}))$ of actual observations and counterfactual interventions, morphisms are pairs of morphisms in **BorelStoch** (one for the actual world, one for the counterfactual) sharing the same exogenous noise distribution. The counterfactual functor $\text{CF} : \mathbf{PossWorlds}(\mathcal{G}) \rightarrow \mathbf{BorelStoch}$ computes the counterfactual distribution in each possible world.

Proposition 25.5 (Modularity of Counterfactuals). *The counterfactual functor CF is natural with respect to the modular structure of the SCM: for any variable V_j not on the causal path from X to Y (not a descendant of X or an ancestor of Y), the counterfactual value $V_{j,x'}$ (the value of V_j in the counterfactual world where $X = x'$) equals its actual value $V_j = v_j$. This modularity is the naturality condition on CF : it commutes with the “projection onto the actual value of V_j ” morphism in **BorelStoch**.*

25.5 Causal Representation Learning and the Double Para Construction

Causal representation learning (Schölkopf et al., 2021) seeks to learn representations of data that respect the underlying causal structure, enabling better generalisation to distribution shifts and interventional distributions. The categorical framework makes precise what it means for a representation to be “causal”.

A *causal encoder* is a morphism $e_\phi : X \rightarrow Z$ in **Para**(**BorelStoch**) such that the pushforward $e_{\phi*}\mu_X$ (the representation distribution) satisfies the causal Markov condition with respect to the true causal graph $\mathcal{G}$. Specifically, the representation $Z = e_\phi(X)$ should factor as $Z = (Z_1, \dots, Z_k)$ (one component per causal variable), and the joint distribution of $(Z_1, \dots, Z_k)$ should factorise as:

$$p(Z_1, \dots, Z_k) = \prod_{i=1}^k p(Z_i \mid Z_{\text{Pa}(i)}),$$

which is the causal Markov condition applied to the representation.

The training of a causal encoder is a morphism in the *double Para construction* $\mathbf{Para}(\mathbf{Para}(\mathbf{BorelStoch}))$: the inner Para captures the task-specific adaptation (learning the representation on a single distribution), and the outer Para captures the meta-causal objective (ensuring the representation satisfies the causal Markov condition across all distributions). The double Para construction corresponds, in the meta-learning context (Chapter 26), to the outer loop of the meta-learning algorithm; in the causal learning context, it corresponds to the outer loop of invariant risk minimisation (IRM; Arjovsky et al., 2019).

Chapter 26

Meta-Learning, MAML, and Higher-Order Categorical Structures

26.1 The Meta-Learning Problem as a 2-Categorical Optimisation

Meta-learning, or “learning to learn”, is the problem of designing learning algorithms that improve with experience across a distribution of tasks, rather than learning each task from scratch. The prototypical algorithms are:

- **MAML** (Model-Agnostic Meta-Learning; Finn, Abbeel, Levine, 2017): find an initialisation θ^* from which fast adaptation (few gradient steps) achieves good performance on any task.
- **Prototypical Networks** (Snell, Swersky, Zemel, 2017): learn an embedding space in which classes are well-separated, enabling nearest-neighbour classification from few examples.
- **Meta-SGD** (Li et al., 2017): learn not just the initialisation but also the per-parameter learning rate.
- **Reptile** (Nichol, Achiam, Schulman, 2018): a first-order approximation to MAML that is more computationally efficient.

The categorical framework for meta-learning uses the 2-categorical structure of **Para(Smooth)**.

Recall that **Para(Smooth)** has:

- Objects: smooth manifolds A .
- 1-morphisms $(P, f) : A \rightarrow B$: pairs of a parameter space P and a smooth map $f : P \times A \rightarrow B$.
- 2-morphisms $\phi : (P, f) \Rightarrow (Q, g)$ (both from A to B): smooth maps $\phi : P \rightarrow Q$ such that $g \circ (\phi \times \text{id}_A) = f$ (reparametrisations making the two 1-morphisms equal).

A meta-learner is a functor $\mathcal{M} : \mathbf{Tasks} \rightarrow \mathbf{Para(Smooth)}$ from the category of tasks to **Para(Smooth)**, together with a distinguished 1-morphism $(\Theta, f_\Theta) : \{*\} \rightarrow \Theta$ (the

initialisation) and a family of 2-morphisms $\{\alpha_\tau : (\Theta, f_\Theta) \Rightarrow (\Theta, f_{\theta_\tau})\}_\tau$ (the adaptation to each task τ , expressed as a reparametrisation of the initialisation). The meta-objective is to find the initialisation that maximises the expected quality of the adaptation:

$$J(\Theta) = \mathbb{E}_{\tau \sim p(\mathcal{T})}[\mathcal{L}_\tau(\theta_\tau)] = \mathbb{E}_{\tau \sim p(\mathcal{T})}[\mathcal{L}_\tau(\alpha_\tau(\Theta))].$$

In MAML, the adaptation α_τ is a fixed number K of gradient steps on the task loss:

$$\alpha_\tau(\theta) = \theta - \sum_{k=0}^{K-1} \alpha \nabla_{\theta} \mathcal{L}_{\tau,k}(\theta_k), \quad \theta_0 = \theta, \quad \theta_{k+1} = \theta_k - \alpha \nabla_{\theta} \mathcal{L}_{\tau,k}(\theta_k),$$

where $\alpha > 0$ is the inner learning rate and $\mathcal{L}_{\tau,k}$ is the task loss on the k -th support batch. Each gradient step $\theta_k \mapsto \theta_{k+1}$ is a 1-morphism in **Para(Smooth)**, and the composition of K steps is a 1-morphism $\alpha_\tau : \Theta \rightarrow \Theta$. The adaptation α_τ is the K -fold composition of 1-morphisms in the 2-category **Para(Smooth)**.

The meta-objective $J(\Theta) = \mathbb{E}_\tau[\mathcal{L}_\tau(\alpha_\tau(\Theta))]$ is a morphism $J : \Theta \rightarrow \mathbb{R}$ in **Smooth**, and the outer optimisation $\theta^* = \arg \min_{\Theta} J(\Theta)$ is a problem in **Para(Smooth)** (find the parameter that minimises J). The gradient of J with respect to Θ requires differentiating through the inner adaptation α_τ , which involves the Hessian of $\mathcal{L}_\tau$.

26.2 The Second-Order Backpropagation Functor

MAML requires computing $\nabla_{\Theta} J(\Theta) = \mathbb{E}_\tau[(D\alpha_\tau(\Theta))^\top \nabla_{\theta_\tau} \mathcal{L}_\tau]$, where $D\alpha_\tau(\Theta)$ is the Jacobian of the adaptation map with respect to the initialisation. For $K = 1$ inner step, $\alpha_\tau(\theta) = \theta - \alpha \nabla_{\theta} \mathcal{L}_\tau(\theta)$, so $D\alpha_\tau(\theta) = I - \alpha \nabla_{\theta}^2 \mathcal{L}_\tau(\theta)$ — the Hessian of the task loss appears.

The categorical account is: the meta-gradient computation requires applying the backpropagation functor **BP** *twice*: once to differentiate through the inner task training (computing the inner gradient $\nabla_{\theta} \mathcal{L}_\tau$), and once to differentiate through the inner gradient computation itself (computing the Hessian $\nabla_{\theta}^2 \mathcal{L}_\tau$). The result is the *second-order backpropagation functor*:

$$\text{BP}_2 = \text{BP} \circ \text{BP} : \mathbf{Para}(\mathbf{Para}(\mathbf{Smooth})) \rightarrow \mathbf{Lens}(\mathbf{Lens}(\mathbf{Smooth})).$$

The 2-categorical structure clarifies what this means. In **Para(Smooth)**, the inner adaptation α_τ is a 1-morphism. In **Para(Para(Smooth))** (the Para construction applied twice), the meta-objective J is a “2-morphism” (a morphism between 1-morphisms), and the meta-gradient is obtained by differentiating this 2-morphism. The functor BP_2 sends:

- The inner loop: $((\Theta, \mathcal{L}_\tau), \alpha_\tau)$ in **Para(Para(Smooth))** $\mapsto$ the lens $(\alpha_\tau, \nabla_{\Theta} \mathcal{L}_\tau \circ D\alpha_\tau)$ in **Lens(Lens(Smooth))**.
- The outer loop: the lens structure gives the meta-gradient $\nabla_{\Theta} J = \mathbb{E}_\tau[\nabla_{\Theta} \mathcal{L}_\tau \circ D\alpha_\tau]$.

The four quantities computed by BP_2 at a single task τ and initialisation Θ are: (1) the adapted parameters $\theta_\tau = \alpha_\tau(\Theta)$; (2) the task loss $\mathcal{L}_\tau(\theta_\tau)$; (3) the inner gradient $\nabla_{\Theta} \mathcal{L}_\tau(\theta_\tau)$ (via the chain rule through α_τ , involving the Hessian); (4) the meta-gradient $\nabla_{\Theta} J(\Theta)$.

First-Order Approximations

First-Order MAML (FOMAML) approximates $D\alpha_\tau(\Theta) \approx I$ (ignoring the Hessian), giving:

$$\nabla_\Theta J^{\text{FO}}(\Theta) \approx \mathbb{E}_\tau[\nabla_{\theta_\tau} \mathcal{L}_\tau(\theta_\tau)].$$

Categorically, FOMAML replaces the 2-functor BP_2 by the 1-functor BP applied only to the outer loop (ignoring the 2-morphism structure of the inner adaptation). The approximation error is $O(\alpha \|\nabla_\theta^2 \mathcal{L}_\tau\|)$: it is small when the inner learning rate α is small or the task loss is nearly linear (small Hessian).

Reptile (Nichol et al., 2018) uses the update $\Theta \leftarrow \Theta + \eta(\theta_\tau - \Theta)$, where θ_τ is obtained by K inner steps. The Reptile update is the average of the task-specific parameter differences, which approximates the meta-gradient in a different sense: it is the gradient of the *average task loss at the adapted parameters*, rather than the meta-gradient of the *task loss at the initialisation*. Categorically, Reptile replaces the 2-functor BP_2 by the 0-functor (evaluation at a fixed point), making it the coarsest possible approximation.

26.3 Few-Shot Learning and the Categorical Geometry of Class Representations

Few-shot learning is the specialisation of meta-learning to classification tasks where each task consists of a small *support set* (a few labelled examples per class) and a larger *query set* (unlabelled examples to classify). The N -way K -shot problem has N classes and K support examples per class.

The Yoneda lemma (Chapter 8) provides a deep interpretation of few-shot learning. The support set $\mathcal{S}_c = \{(x_i^c, c)\}_{i=1}^K$ for class c represents the “empirical Yoneda representation” of c : the class is described by its relationships (feature similarities) to the K support examples. The classification of a query q is then:

$$\hat{c}(q) = \arg \min_c d(f_\phi(q), p_c),$$

where $p_c = \frac{1}{K} \sum_i f_\phi(x_i^c)$ is the class prototype (the mean of the Yoneda representation) and d is a distance in the embedding space.

Theorem 26.1 (Optimality of Prototypical Classifiers, Categorical Form). *Let $f_\phi : X \rightarrow \mathbb{R}^d$ be an encoder and p_c the class prototype. Under a Gaussian generative model $p(x | c) = \mathcal{N}(f_\phi(x); \mu_c, \sigma^2 I)$ with equal covariance, the prototypical classifier $\hat{c}(q) = \arg \min_c \|f_\phi(q) - p_c\|^2$ is the Bayes-optimal classifier. Categorically: the prototype $p_c = \mathbb{E}[f_\phi(X) | C = c]$ is the mean of the Yoneda representation of class c in the embedding space, and the nearest-prototype rule is the maximum a posteriori (MAP) rule in the Markov category **BorelStoch** under the Gaussian model.*

The Yoneda lemma then guarantees optimality: if the encoder f_ϕ satisfies $\text{Nat}(\mathcal{C}(c, -), F_\phi) \cong F_\phi(c)$ (the Yoneda condition: the natural transformations from the representable functor of c to the feature functor F_ϕ are in bijection with the features of c), then the prototypical

classifier achieves the minimum expected 0-1 loss among all classifiers that use only the support set prototypes.

26.4 Hypernetworks as Morphisms in the Internal Category

A *hypernetwork* (Ha, Dai, Le, 2017) is a network $h_\psi : Z \rightarrow \Theta$ that generates the weights of a target network $f_\theta : X \rightarrow Y$, given a task embedding $z \in Z$. The composed system $(z, x) \mapsto f_{h_\psi(z)}(x)$ adapts the target network to each task by generating task-specific weights.

The categorical account uses the *internal hom* and the *internal category* of **Para(Smooth)**. In a cartesian closed category (CCC), the internal hom $[A, B]$ represents the space of all morphisms from A to B . The Para construction adds parametrisation: **Para(Smooth)** $([A, B], -)$ is the functor that sends a parameter space P to the set of parametrised maps $P \times A \rightarrow B$.

A hypernetwork is a morphism $h_\psi : Z \rightarrow \text{Mor}(\mathbf{Para}(\mathbf{Smooth}))$ from the task embedding space Z into the “object of morphisms” of the internal category of **Para(Smooth)**:

$$\text{Mor}(\mathbf{Para}(\mathbf{Smooth})) = \bigsqcup_{\theta \in \Theta} \{f_\theta : X \rightarrow Y\},$$

the disjoint union of all parametrised maps. The hypernetwork selects a morphism $f_{h_\psi(z)} : X \rightarrow Y$ for each task embedding z , and evaluation gives the composed map $(z, x) \mapsto f_{h_\psi(z)}(x) = \text{ev}(h_\psi(z), x)$, where $\text{ev} : \text{Mor}(\mathbf{Para}(\mathbf{Smooth})) \times X \rightarrow Y$ is the internal evaluation morphism.

This is the categorical self-reference of the Para construction: the hypernetwork maps task embeddings to morphisms in Para, making the Para construction itself a “parametrised morphism space”. The training of the hypernetwork is a morphism in **Para(Para(Para(Smooth)))** — the triple Para construction — where the outermost Para captures the hypernetwork parameters ψ , the middle Para captures the generated target network parameters $\theta = h_\psi(z)$, and the innermost Para captures the target network’s computation.

Chapter 27

The Categorical Theory of Optimisation

27.1 The Para Construction as an Optimisation Problem

Optimisation in machine learning is the problem of finding the parameter $\theta^* \in \Theta$ that minimises a loss function $\mathcal{L} : \Theta \rightarrow \mathbb{R}$ (or, in the stochastic setting, maximises an expected reward). In the categorical framework, both the parameter space Θ and the loss $\mathcal{L}$ are morphisms in specific categories, and the optimisation algorithm is a dynamical system (Chapter 5) on Θ .

Formally, an optimisation algorithm Opt for a differentiable loss $\mathcal{L} : \Theta \rightarrow \mathbb{R}$ is a morphism:

$$\text{Opt} : \Theta \times T^*\Theta \rightarrow \Theta \quad \text{in } \mathbf{Smooth},$$

taking the current parameter θ and the gradient $\nabla_\theta \mathcal{L}(\theta) \in T_\theta^*\Theta$ (a covector in the cotangent space), and producing the updated parameter. The *training loop* iterates this morphism n times over mini-batches of data, giving a morphism $\text{Opt}^n : \Theta \times (T^*\Theta)^n \rightarrow \Theta$ in **Smooth**.

As observed in Chapter 5, this is precisely a dynamical system in the category **Poly** of polynomial functors: the state space is Θ , the input space is $T^*\Theta$ (the gradient), the output space is Θ (the updated parameters), and the update map is Opt . The training loop for n steps is the n -fold iteration of this dynamical system, and the convergence of training is the approach of this dynamical system to its attractor (the set of local minima of $\mathcal{L}$).

27.2 The Geometry of Gradient Descent

Vanilla gradient descent with learning rate $\eta > 0$ is the update:

$$\text{GD}(\theta, g) = \theta - \eta g, \quad g = \nabla_\theta \mathcal{L}(\theta).$$

This is a linear morphism in **FVect** $_{\mathbb{R}}$ (for a flat parameter space $\Theta = \mathbb{R}^d$), and the training loop is the linear iteration $\theta_{t+1} = (I - \eta H)\theta_t$ (for a quadratic loss $\mathcal{L}(\theta) = \frac{1}{2}\theta^\top H\theta + b^\top \theta + c$,

where $H = \nabla^2 \mathcal{L}$ is the Hessian). The convergence rate is determined by the spectral radius $\rho(I - \eta H) = \max_i |1 - \eta \lambda_i(H)|$, and the optimal learning rate is $\eta^* = 2/(\lambda_{\min} + \lambda_{\max})$, giving convergence rate $\kappa = (\lambda_{\max} - \lambda_{\min})/(\lambda_{\max} + \lambda_{\min})$ (the condition number ratio).

The continuous-time limit ($\eta \rightarrow 0$) gives the *gradient flow ODE*:

$$\dot{\theta}(t) = -\nabla_{\theta} \mathcal{L}(\theta(t)),$$

a morphism in **Para(ODE)** (Chapter 24). The gradient flow is the steepest descent of $\mathcal{L}$ in the Euclidean metric on Θ : it is the negative of the Riemannian gradient of $\mathcal{L}$ with respect to the flat metric $\langle \cdot, \cdot \rangle_{\Theta}$. The natural gradient (Chapter 19) replaces the Euclidean metric with the Fisher-Rao metric, giving the steepest descent in the statistical manifold:

$$\dot{\theta}(t) = -\mathcal{I}(\theta(t))^{-1} \nabla_{\theta} \mathcal{L}(\theta(t)),$$

which is invariant under reparametrisation of Θ (a consequence of the naturality of the musical isomorphism $\mathcal{I}^{-1} : T^* \Theta \rightarrow T \Theta$).

27.3 Stochastic Gradient Descent and the Langevin Equation

Stochastic gradient descent (SGD) replaces the full gradient $\nabla_{\theta} \mathcal{L}$ with a mini-batch gradient $\hat{g}_t = \nabla_{\theta} \mathcal{L}_{B_t}(\theta)$ (computed on a random mini-batch B_t of the data). The mini-batch gradient is an unbiased estimator of the full gradient: $\mathbb{E}[\hat{g}_t] = \nabla_{\theta} \mathcal{L}$, with variance $\text{Var}[\hat{g}_t] \propto \sigma^2/|B_t|$.

In the continuous-time limit with noise, the SGD iterate approximates the *Langevin SDE*:

$$d\theta_t = -\nabla_{\theta} \mathcal{L}(\theta_t) dt + \sqrt{2T} dW_t,$$

where $T \propto \eta/|B|$ is the effective temperature (proportional to the learning rate and inversely proportional to batch size), and W_t is standard Brownian motion. This SDE is a morphism in **SDE** on the parameter manifold Θ , and its stationary distribution is the Gibbs measure:

$$\pi^*(\theta) \propto \exp\left(-\frac{\mathcal{L}(\theta)}{T}\right).$$

The Fokker-Planck equation for the density ρ_t (the distribution over parameters at time t) is:

$$\partial_t \rho_t = \nabla \cdot (\rho_t \nabla \mathcal{L}) + T \Delta \rho_t.$$

This PDE can be rewritten as the *Wasserstein gradient flow*:

$$\partial_t \rho_t = -\nabla_{W_2} \mathcal{F}[\rho_t], \quad \mathcal{F}[\rho] = \int \mathcal{L}(\theta) \rho(\theta) d\theta + T \int \rho(\theta) \log \rho(\theta) d\theta,$$

where ∇_{W_2} denotes the gradient in the Wasserstein-2 metric on $\mathcal{G}(\Theta)$. The free energy functional $\mathcal{F}[\rho]$ has the Gibbs measure π^* as its unique minimiser, so the Langevin dynamics converges to π^* at rate $e^{-\rho_{\text{LSI}} t}$ (where ρ_{LSI} is the log-Sobolev inequality constant of π^*).

Categorically, the Wasserstein gradient flow is a morphism in the enriched category of probability measures on Θ (with the Wasserstein-2 metric as the enriched hom, Chapter 7), and its convergence is the statement that the gradient flow in this enriched category has π^* as its unique attractor (a fixed-point theorem in the Lawvere metric category).

27.4 Momentum Methods and Second-Order Geometry

The *heavy ball method* (Polyak, 1964) augments gradient descent with a momentum term:

$$v_{t+1} = \beta v_t - \eta \nabla \mathcal{L}(\theta_t), \quad \theta_{t+1} = \theta_t + v_{t+1},$$

where $\beta \in [0, 1)$ is the momentum coefficient. The update rule for $(\theta_t, v_t) \in T\Theta = \Theta \times T_\theta\Theta$ (position and velocity in parameter space) is a morphism $\text{Mom} : T\Theta \times T^*\Theta \rightarrow T\Theta$ in **Smooth**.

In continuous time, the heavy ball iteration converges to the *underdamped harmonic oscillator ODE*:

$$\ddot{\theta} + \gamma \dot{\theta} + \nabla \mathcal{L}(\theta) = 0,$$

where $\gamma = (1 - \beta)/\eta$ is the friction coefficient. This is a second-order ODE on Θ , which lifts to a first-order system on $T\Theta$:

$$(d\theta/dt, dv/dt) = (v, -\gamma v - \nabla \mathcal{L}(\theta)).$$

The vector field $\mathbf{F}(\theta, v) = (v, -\gamma v - \nabla \mathcal{L}(\theta))$ defines a morphism in **Para(ODE)** on the tangent bundle $T\Theta$.

The stochastic version (underdamped Langevin dynamics):

$$d\theta = v dt, \quad dv = (-\gamma v - \nabla \mathcal{L}(\theta)) dt + \sqrt{2\gamma T} dW,$$

has stationary distribution $\pi^*(\theta, v) \propto \exp(-(\mathcal{L}(\theta) + \|v\|^2/2)/T)$ — the kinetic Gibbs measure on phase space $T\Theta$. The marginal over θ is the same Gibbs measure $\pi^*(\theta) \propto e^{-\mathcal{L}(\theta)/T}$, so the underdamped Langevin samples from the correct distribution. The key advantage is mixing speed: underdamped Langevin mixes faster than overdamped Langevin (vanilla SGD + noise) by a factor of $O(\sqrt{\kappa})$ where κ is the condition number, because the momentum allows the dynamics to traverse the loss landscape more efficiently.

The convergence analysis of underdamped Langevin uses Villani's *hypocoercivity* theory (Villani, 2009): although noise enters only through the velocity v (the diffusion is degenerate), the combined effect of the deterministic drift and the stochastic noise in velocity covers all directions in $T\Theta$ (hypoellipticity), ensuring exponential convergence to π^* .

27.5 Adam, Preconditioned Gradient Descent, and the Fisher Metric

The *Adam* optimiser (Kingma-Ba, 2015) maintains exponential moving averages of the gradient and the squared gradient:

$$\begin{aligned}
 m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t && \text{(first moment: gradient EMA),} \\
 v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 && \text{(second moment: squared gradient EMA),} \\
 \hat{m}_t &= m_t / (1 - \beta_1^t), \quad \hat{v}_t = v_t / (1 - \beta_2^t) && \text{(bias correction),} \\
 \theta_{t+1} &= \theta_t - \eta \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon). && \text{(parameter update)}
 \end{aligned}$$

The update $\hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ is an element-wise preconditioned gradient: each coordinate i of the gradient is scaled by $1 / (\sqrt{\hat{v}_t^i} + \varepsilon) \approx 1 / \text{rms}(g_t^i)$, where rms is the root mean square. This gives each parameter an effective learning rate $\eta / \text{rms}(g_t^i)$ that is large when the gradient variance is small (confident, stable gradients) and small when the gradient variance is large (noisy, uncertain gradients).

Proposition 27.1 (Adam as Diagonal Natural Gradient). *The Adam update $\hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ approximates the diagonal natural gradient: $\mathcal{I}_{\text{diag}}(\theta)^{-1} \nabla_{\theta} \mathcal{L}$, where $\mathcal{I}_{\text{diag}}(\theta) = \text{diag}(\mathbb{E}[g^2]) \approx \text{diag}(\hat{v}_t)$ is the diagonal of the Fisher information matrix. The bias-corrected EMA $\hat{v}_t \rightarrow \mathbb{E}[g_t^2]$ as $t \rightarrow \infty$ (assuming stationary gradient statistics), so Adam converges to the diagonal natural gradient in the large- t limit.*

The categorical structure: Adam is a morphism $\text{Adam} : (\Theta \times \mathbb{R}^d \times \mathbb{R}_{>0}^d) \times T^* \Theta \rightarrow \Theta \times \mathbb{R}^d \times \mathbb{R}_{>0}^d$ in **Smooth**, where the state includes the parameter θ , first moment m , and second moment v . The state space $\Theta \times \mathbb{R}^d \times \mathbb{R}_{>0}^d$ is the total space of the trivial vector bundle $\Theta \times \mathbb{R}^{2d}$, and the Adam update is a smooth section of this bundle.

The bias-correction factors $(1 - \beta_1^t)^{-1}$ and $(1 - \beta_2^t)^{-1}$ are natural transformations from the EMA functor to the identity functor: they correct the initialisation bias (since $m_0 = v_0 = 0$ biases the first few estimates toward zero). Categorically, the bias correction is the normalisation of the Kleisli morphism for the exponential moving average monad (Chapter 13) at early times, ensuring that the morphism converges to the true gradient statistics.

More precise approximations to the natural gradient — K-FAC (Martens-Grosse, 2015), EK-FAC, Shampoo (Gupta et al., 2018), distributed Shampoo — use block-diagonal, Kronecker-factored, or full-matrix approximations to $\mathcal{I}(\theta)^{-1}$. Each is a morphism in a subcategory of the category of preconditioned gradient methods, with the precision of the approximation determined by how closely the preconditioner approximates the true Fisher inverse.

27.6 The Loss Landscape: Saddle Points, Flat Minima, and the Morse-Theoretic Account

The loss landscape $\mathcal{L} : \Theta \rightarrow \mathbb{R}$ is a smooth function on the parameter manifold. Its critical points (where $\nabla \mathcal{L}(\theta) = 0$) are classified by their *Morse index* (the number of negative

eigenvalues of the Hessian $\nabla^2 \mathcal{L}(\theta)$): index 0 = local minimum, index d = local maximum, index $1, \dots, d-1$ = saddle points.

For large neural networks in the overparametrised regime ($\dim \Theta \gg \text{training data}$), the loss landscape has a specific structure (predicted by random matrix theory, Chapter 11):

- Most critical points are saddle points (not minima or maxima).
- The loss value at a critical point is correlated with its Morse index: higher loss $\Rightarrow$ more negative eigenvalues (more directions of descent).
- There are exponentially many local minima with approximately equal loss (the degenerate minima form a connected manifold in Θ).

The Morse complex of $\mathcal{L}$ (the CW complex built from the stable manifolds of the gradient flow) has Betti numbers determined by these critical points, and the Morse homology $H_k(\Theta; \mathcal{L}) = \ker \partial_k / \text{im } \partial_{k+1}$ (where ∂_k counts gradient flow connections between index- k and index- $(k+1)$ critical points) is isomorphic to the singular homology of Θ .

For neural network training, the practical implication is that gradient descent avoids most saddle points (because the gradient points away from the saddle in the directions of negative curvature) and converges to one of the many nearly-equivalent local minima. The *flat minima hypothesis* (Hochreiter-Schmidhuber, 1997; Keskar et al., 2017) states that minima with small Hessian eigenvalues (flat minima) generalise better than minima with large eigenvalues (sharp minima), and this is supported by the PAC-Bayes analysis (Chapter 22): the complexity term $\text{KL}(Q\|P)$ in the PAC-Bayes bound is smaller for flat minima (where a large ball of parameters achieves low loss) than for sharp minima (where only a small ball achieves low loss).

Sharpness-Aware Minimisation (SAM; Foret et al., 2021) directly minimises the worst-case loss in a ball around the current parameters:

$$\min_{\theta} \max_{\|\varepsilon\|_2 \leq \rho} \mathcal{L}(\theta + \varepsilon) \approx \min_{\theta} \mathcal{L}(\theta + \rho \hat{g}) + \lambda \|\hat{g}\|^2,$$

where $\hat{g} = \nabla \mathcal{L}(\theta) / \|\nabla \mathcal{L}(\theta)\|$ is the normalised gradient. SAM is an open game (Chapter 10): the outer player (parameter) minimises and the inner player (perturbation) maximises. The Nash equilibrium of this game is a parameter θ^* satisfying:

- Inner equilibrium: $\varepsilon^* = \rho \nabla \mathcal{L}(\theta^*) / \|\nabla \mathcal{L}(\theta^*)\|$ (gradient ascent on the perturbation).
- Outer equilibrium: θ^* minimises $\mathcal{L}(\theta + \varepsilon^*(\theta))$ (gradient descent on the perturbed loss).

At equilibrium, θ^* is a flat minimum: the worst-case perturbation in direction $\hat{g}$ does not increase the loss much, implying that $\lambda_{\max}(\nabla^2 \mathcal{L}(\theta^*))$ is small (the maximum sharpness is bounded).

The categorical connection: the SAM game is a morphism in **Game** (the category of open games, Chapter 10), and the Nash equilibrium is the categorical fixed point of the game composition. The PAC-Bayes analysis (Chapter 22) bounds the generalisation gap by $\text{KL}(Q\|P) + \log(m/\delta)$, where $Q = \mathcal{N}(\theta^*, \sigma^2 I)$ is the posterior concentrated at the SAM

minimum and $P = \mathcal{N}(0, \sigma^2 I)$ is the prior; for flat minima, the KL is small (the prior and posterior are close), tightening the bound. This unified categorical account of SAM connects the game-theoretic structure (open games), the information-geometric structure (Fisher metric and natural gradient), and the PAC-Bayes generalisation theory into a single coherent framework.

Chapter 28

Graph Neural Networks and the Categorical Geometry of Relational Data

28.1 The Category of Graphs

Graphs are among the most fundamental relational structures in mathematics and in applied science. A molecule is a graph whose vertices are atoms and whose edges are chemical bonds. A social network is a graph of people and friendships. A knowledge graph represents entities and semantic relations. A protein interaction network, a road network, a parse tree, a program control-flow graph — all are graphs. The central challenge of graph-based machine learning is to compute functions on graphs that are *intrinsic*: that depend on the graph structure and the data attached to it, but not on any arbitrary labelling or ordering of the vertices and edges.

We begin by establishing the categorical framework for graphs. A *directed multigraph* (or *quiver*) is a tuple (V, E, s, t) where V is the vertex set, E is the edge set, and $s, t : E \rightarrow V$ are the source and target maps. The category **Quiv** has quivers as objects and *quiver morphisms* (ϕ_V, ϕ_E) (pairs of functions satisfying $\phi_V \circ s = s' \circ \phi_E$ and $\phi_V \circ t = t' \circ \phi_E$) as morphisms.

The critical observation, which opens the door to the full power of category theory, is that **Quiv** is a presheaf category. Define the *walking quiver* category **Q** with two objects V and E and two non-identity morphisms $s, t : E \rightarrow V$. Then there is a canonical equivalence of categories:

$$\mathbf{Quiv} \simeq [\mathbf{Q}^{\text{op}}, \mathbf{Set}],$$

where $[\mathbf{Q}^{\text{op}}, \mathbf{Set}]$ is the functor category of presheaves on **Q**. A presheaf $F : \mathbf{Q}^{\text{op}} \rightarrow \mathbf{Set}$ assigns to the object V the vertex set $F(V)$ and to the object E the edge set $F(E)$, and to the morphisms s, t the functions $F(s), F(t) : F(E) \rightarrow F(V)$ — exactly the data of a quiver. A natural transformation between two such presheaves is a quiver morphism.

This identification carries major consequences. Since $[\mathbf{Q}^{\text{op}}, \mathbf{Set}]$ is a Grothendieck topos

(Chapter 9), the category $\mathbf{Quiv}$ inherits all the logical and colimit-completion structure of $\mathbf{topoi}$: it has all limits and colimits (computed pointwise), a subobject classifier, an internal language, and the full Yoneda machinery. The Yoneda embedding $\mathcal{Y} : \mathbf{Q} \rightarrow \mathbf{Quiv}$ sends the object V to the representable presheaf $\mathbf{Q}(-, V)$, which is the “single vertex, no edges” quiver, and E to the “single edge” quiver (two vertices and one directed edge between them). Every quiver is a colimit of copies of these two basic quivers, a fact that underlies the universal approximation results for graph neural networks.

For *undirected* graphs we impose a symmetry: an involution $\iota : E \rightarrow E$ with $\iota^2 = \text{id}_E$, $s \circ \iota = t$, and $t \circ \iota = s$ (every edge has a reverse). The category $\mathbf{Grph}$ of undirected graphs and graph morphisms is a reflective subcategory of $\mathbf{Quiv}$: the inclusion $\mathbf{Grph} \hookrightarrow \mathbf{Quiv}$ has a left adjoint (symmetrisation functor) and a right adjoint (underlying undirected graph functor).

28.2 Attributed Graphs and the Category $\mathbf{Grph}_{\mathcal{F}}$

In machine learning, graphs carry data. Each vertex v has a feature vector $x_v \in \mathbb{R}^{d_V}$, each edge (u, v) has a feature vector $e_{uv} \in \mathbb{R}^{d_E}$, and possibly the entire graph has a global feature $u \in \mathbb{R}^{d_G}$. An *attributed graph* is a pair (G, x) where $G = (V, E, s, t)$ is a graph and $x = (x_V : V \rightarrow \mathbb{R}^{d_V}, x_E : E \rightarrow \mathbb{R}^{d_E})$ is a feature assignment.

The category $\mathbf{Grph}_{\mathcal{F}}$ of attributed graphs has:

- *Objects*: pairs (G, x) .
- *Morphisms*: pairs (ϕ, L) where $\phi : G \rightarrow G'$ is a graph morphism and $L = (L_V : \mathbb{R}^{d_V} \rightarrow \mathbb{R}^{d'_V}, L_E : \mathbb{R}^{d_E} \rightarrow \mathbb{R}^{d'_E})$ are linear maps satisfying $x'_V(\phi_V(v)) = L_V(x_V(v))$ for all $v \in V$ and $x'_E(\phi_E(e)) = L_E(x_E(e))$ for all $e \in E$.

This is the comma category $(\mathbf{Grph} \downarrow \mathcal{F})$, where $\mathcal{F} : \mathbf{Grph} \rightarrow \mathbf{FVect}_{\mathbb{R}}$ is the functor sending each graph to its feature space. The morphism condition expresses that a graph morphism must simultaneously be structure-preserving (on the graph) and linearly compatible (on the features).

A *graph neural network* (GNN) with L layers is a morphism in $\mathbf{Para}(\mathbf{Grph}_{\mathcal{F}})$: a pair $((\Theta_1, \dots, \Theta_L), f)$ where the parameter space is the product of the layer parameter spaces and the implementation $f : (\Theta_1 \times \dots \times \Theta_L) \times (G, x) \rightarrow (G, h)$ maps each attributed graph to the same graph with updated features. The backpropagation functor $\mathbf{BP} : \mathbf{Para}(\mathbf{Grph}_{\mathcal{F}}) \rightarrow \mathbf{Lens}(\mathbf{Grph}_{\mathcal{F}})$ gives the gradient computation, with the forward pass computing updated features and the backward pass propagating gradients from the output through each layer.

28.3 Message Passing as a Natural Transformation

The message-passing neural network (MPNN) framework (Gilmer et al., 2017) subsumes the majority of GNN architectures in a single scheme. The k -th layer update rule is:

$$\begin{aligned} m_v^{(k)} &= \mathbf{Agg}\left(\{M_\theta(h_v^{(k-1)}, h_u^{(k-1)}, e_{vu}) : u \in \mathcal{N}(v)\}\right), \\ h_v^{(k)} &= U_\theta\left(h_v^{(k-1)}, m_v^{(k)}\right), \end{aligned}$$

where M_θ is the *message function*, $\mathbf{Agg}$ is a permutation-invariant *aggregation function* (sum, mean, or max), and U_θ is the *update function*. All three are neural networks with parameters in Θ .

The categorical interpretation is clean. Define the *neighbourhood functor*

$$\mathcal{N} : \mathbf{Grph}_{\mathcal{F}} \rightarrow \mathbf{Grph}_{\mathcal{MS}},$$

where $\mathbf{Grph}_{\mathcal{MS}}$ is the category of graphs with multiset-valued features, sending each attributed graph (G, h) to the attributed graph where each vertex v is endowed with the multiset $\mathcal{N}(v, h) = \{\{h_u : u \in \mathcal{N}(v)\}\}$ of its neighbours' features. The MPNN update is then the composite natural transformation:

$$\mathbf{MPNN}_\theta = U_\theta \circ \mathbf{Agg} \circ M_\theta \circ \mathcal{N} : \mathbf{Id}_{\mathbf{Grph}_{\mathcal{F}}} \Rightarrow \mathbf{Id}_{\mathbf{Grph}_{\mathcal{F}}}.$$

Proposition 28.1 (Equivariance as Naturality). *The MPNN layer $\mathbf{MPNN}_\theta$ is a natural transformation in $\mathbf{Grph}_{\mathcal{F}}$ if and only if: (1) the message function M_θ is equivariant under permutations of the edge endpoints; (2) the aggregation $\mathbf{Agg}$ is a symmetric function on multisets; (3) the update U_θ is applied identically at each vertex. These three conditions are precisely the definition of a message-passing layer. Under these conditions, for any graph isomorphism $\phi : (G, h) \rightarrow (G', h')$,*

$$\mathbf{MPNN}_\theta(G', h') = (\phi, \text{id}) \circ \mathbf{MPNN}_\theta(G, h) \circ (\phi^{-1}, \text{id}),$$

i.e., permuting the input vertices permutes the output features by the same permutation. This is permutation equivariance.

Proof. The naturality square for ϕ requires commutativity of

$$\begin{array}{ccc} (G, h) & \xrightarrow{\mathbf{MPNN}_\theta} & (G, h') \\ \phi \downarrow & & \downarrow \phi \\ (G', \phi_* h) & \xrightarrow{\mathbf{MPNN}_\theta} & (G', \phi_* h') \end{array}$$

The neighbourhood functor satisfies $\mathcal{N}(\phi_* h)_{\phi(v)} = \phi_*(\mathcal{N}(h)_v)$ (neighbours are preserved by ϕ). The message function M_θ applied to $\phi_* h$ at vertex $\phi(v)$ gives $M_\theta(\phi(h_v), \phi(h_u), e_{\phi(v)\phi(u)}) = \phi_*(M_\theta(h_v, h_u, e_{vu}))$ by condition (1). Aggregation over the multiset commutes with the push-forward by condition (2), and the update U_θ commutes by condition (3). The entire

composite is natural. $\square$

28.4 The Weisfeiler-Leman Test and the Limits of MPNN Expressivity

The *1-dimensional Weisfeiler-Leman* (1-WL) colour refinement algorithm iteratively assigns colours to vertices: initially all vertices receive the same colour; at each step, a vertex's new colour is the hash of its current colour and the multiset of its neighbours' colours:

$$c_v^{(k+1)} = \text{hash}\left(c_v^{(k)}, \{\{c_u^{(k)} : u \in \mathcal{N}(v)\}\}\right).$$

The algorithm terminates when the colouring stabilises. Two graphs are declared non-isomorphic if their stable colour multisets differ.

Theorem 28.2 (Xu et al., 2019; categorical form). *Let $f_\theta : \mathbf{Grph}_{\mathcal{F}} \rightarrow \mathbb{R}^k$ be a GNN with sum aggregation. The following are equivalent:*

1. f_θ is maximally expressive (distinguishes all graph pairs that 1-WL can distinguish).
2. The map $v \mapsto h_v^{(\infty)}$ (stable representation after all layers) is injective on the set of rooted subgraphs of any graph in the domain.
3. Each MPNN layer update $h_v^{(k+1)} = \phi_\theta((1 + \varepsilon)h_v^{(k)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k)})$ uses an injective function ϕ_θ (e.g., an MLP with sufficient width).

The graph isomorphism network (GIN) achieves maximal 1-WL expressivity by implementing condition (3).

The categorical content: define the *WL equivalence relation* $\sim_{\text{WL}}$ on $\mathbf{Grph}$ by $G \sim_{\text{WL}} G'$ iff the 1-WL algorithm produces the same stable colour multiset. The quotient category $\mathbf{Grph}/\sim_{\text{WL}}$ is the WL quotient; a function $f : \mathbf{Grph} \rightarrow \mathbb{R}^k$ is WL-invariant iff it factors through the quotient map $\pi : \mathbf{Grph} \rightarrow \mathbf{Grph}/\sim_{\text{WL}}$. The GIN is the morphism in $\mathbf{Para}(\mathbf{Grph}_{\mathcal{F}})$ that, for any pair (G, G') with $G \not\sim_{\text{WL}} G'$, produces distinct outputs — i.e., the factored function $\bar{f} : \mathbf{Grph}/\sim_{\text{WL}} \rightarrow \mathbb{R}^k$ is injective.

The *k-WL hierarchy* provides strictly more expressive tests for $k \geq 2$: the k -WL algorithm assigns colours to all k -tuples of vertices and refines them using the $(k+1)$ -tuple neighbourhood structure. A GNN matching k -WL expressivity must operate on k -tuples (with computational cost $O(n^k)$), so there is a fundamental tradeoff between expressivity and efficiency. Each level of the hierarchy corresponds to a finer quotient category of $\mathbf{Grph}$, and the inclusion of quotient categories $\mathbf{Grph}/\sim_{k\text{-WL}} \hookrightarrow \mathbf{Grph}/\sim_{(k-1)\text{-WL}}$ is a strict inclusion (the k -WL test is strictly more powerful than the $(k-1)$ -WL test for all $k \geq 2$).

28.5 Spectral Graph Theory and the Graph Fourier Transform

The spectral approach to GNNs uses the eigenstructure of the graph Laplacian to define graph convolution in analogy with the classical Fourier transform. For an undirected graph

G with n vertices, adjacency matrix $A \in \{0, 1\}^{n \times n}$, and degree matrix $D = \text{diag}(\sum_j A_{ij})$, the *normalised graph Laplacian* is:

$$\mathcal{L} = I - D^{-1/2} A D^{-1/2} = U \Lambda U^\top,$$

with eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n \leq 2$ and orthonormal eigenvectors $U = [u_1, \dots, u_n]$.

The *graph Fourier transform* of a signal $f \in \mathbb{R}^n$ is $\hat{f} = U^\top f$, and graph convolution with a filter $\hat{g} \in \mathbb{R}^n$ (specified in the spectral domain) is:

$$f \star_G g = U(\hat{g} \odot \hat{f}) = U \text{diag}(\hat{g}) U^\top f.$$

Theorem 28.3 (Fourier Transform as Monoidal Functor, Graph Case). *The graph Fourier transform $\mathcal{F}_G : (\mathbb{R}^n, \star_G, \delta_{v_0}) \rightarrow (\mathbb{R}^n, \odot, \mathbf{1})$ is a symmetric monoidal functor from the algebra of graph signals under convolution (with unit the delta function at any vertex v_0) to the algebra of Fourier coefficients under pointwise multiplication (with unit the all-ones vector). The graph convolution theorem $\mathcal{F}_G(f \star_G g) = \mathcal{F}_G(f) \odot \mathcal{F}_G(g)$ is precisely the monoidal functor law for $\mathcal{F}_G$.*

The *Chebyshev spectral GCN* (Defferrard-Bresson-Vanderghelynst, 2016) approximates graph filters as polynomials in the Laplacian:

$$f \star_G g_\theta = \sum_{k=0}^K \theta_k T_k(\tilde{\mathcal{L}}) f,$$

where T_k are Chebyshev polynomials and $\tilde{\mathcal{L}} = 2\mathcal{L}/\lambda_n - I$ is normalised. The first-order approximation ($K = 1$) with $\lambda_n \approx 2$ and $\theta_0 = -\theta_1 = \theta$ gives the vanilla GCN layer (Kipf-Welling, 2017):

$$H^{(\ell+1)} = \sigma\left(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} H^{(\ell)} W^{(\ell)}\right), \quad \hat{A} = A + I, \quad \hat{D}_{ii} = \sum_j \hat{A}_{ij}.$$

The categorical content of the GCN is the following. The normalised adjacency $\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$ is a doubly stochastic matrix — a morphism in the Markov category **BorelStoch** (Chapter 6), defining a random walk on the graph. The GCN layer is the composition of this Markov kernel (spreading information across the graph) with a linear transformation $W^{(\ell)}$ (transforming the features) and a non-linear activation σ . Repeated application of the GCN layer corresponds to iterating the Markov kernel, which converges (under ergodicity conditions on the graph) to the stationary distribution of the random walk. This *over-smoothing* phenomenon — all vertex features converging to the same value in deep GCNs — is the graph-theoretic analogue of the fixed-point theorem for Markov chains (Chapter 29).

28.6 Graph Pooling as a Pushforward in the Markov Category

Graph-level prediction tasks require aggregating vertex features into a single vector representing the entire graph. *Global pooling* computes:

$$h_G = \text{Pool}(\{h_v : v \in V\}).$$

Sum pooling ($h_G = \sum_v h_v$) is a morphism $\text{Pool}_\Sigma : \mathbf{Grph}_{\mathcal{F}} \rightarrow \mathbf{FVect}_{\mathbb{R}}$ sending each attributed graph to the sum of its vertex features. Mean pooling ($h_G = \frac{1}{|V|} \sum_v h_v$) is a morphism $\text{Pool}_\mu : \mathbf{Grph}_{\mathcal{F}} \rightarrow \mathcal{G}(\mathbf{FVect}_{\mathbb{R}})$ (a stochastic morphism, because the mean is the expectation under the uniform distribution over vertices).

Both sum and mean pooling are natural transformations by the same argument as for MPNN equivariance: they commute with graph isomorphisms. The distinction between them is categorical: sum pooling is the colimit (coproduct in $\mathbf{FVect}_{\mathbb{R}}$) of the vertex feature diagram, hence sensitive to graph size; mean pooling is the colimit in the category of probability measures, hence size-invariant.

Hierarchical pooling (DiffPool, MinCutPool) learns a soft assignment matrix $S \in \mathbb{R}^{n \times k}$ (a Markov kernel from n vertices to k clusters) and computes:

$$A' = S^\top A S \in \mathbb{R}^{k \times k}, \quad H' = S^\top H \in \mathbb{R}^{k \times d}.$$

The coarsened graph (A', H') is the pushforward of the original graph along S . Categorically, this is a morphism in $\mathbf{Para}(\mathbf{BorelStoch})$: the soft assignment S is a parametrised Markov kernel (one distribution over clusters per vertex), and the coarsening is its pushforward. The Kan extension interpretation (Chapter 12) holds: $H' = S^\top H$ is the left Kan extension of the vertex feature map along the soft assignment, computing the most information-preserving coarsening consistent with S .

28.7 Graph Transformers and Positional Encodings

Graph Transformers augment the message-passing framework with a global self-attention mechanism: each vertex attends to all other vertices, computing attention weights based on feature similarity rather than graph adjacency. The attention score between vertices u and v is:

$$\alpha_{uv} = \frac{\exp(q_u \cdot k_v / \sqrt{d})}{\sum_w \exp(q_u \cdot k_w / \sqrt{d})}, \quad q_u = h_u W_Q, \quad k_v = h_v W_K,$$

and the attention output is $h'_u = \sum_v \alpha_{uv} (h_v W_V)$.

The attention matrix $\text{Attn} = (\alpha_{uv})_{u,v}$ is a Markov kernel on the vertex set: a morphism in $\mathbf{BorelStoch}$ from vertex indices to distributions over vertex indices. Unlike the fixed Laplacian kernel $\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$ of the GCN, the attention kernel is *parametric* (depends on the current features, hence learned) and *global* (every vertex attends to every other vertex, not just neighbours). Categorically, the fixed Laplacian kernel is a morphism in

BorelStoch, while the attention kernel is a morphism in **Para**(**BorelStoch**).

Since self-attention is permutation equivariant (it has no inherent sense of position), Graph Transformers require *positional encodings* to break the symmetry and inform the model of the graph's geometry. The most common choices are:

Laplacian eigenvector encodings: use the first k eigenvectors of the Laplacian as positional encodings, $\text{PE}(v) = [u_2(v), \dots, u_{k+1}(v)]$. These are equivariant under graph automorphisms (permutations that preserve the graph structure) but not under general permutations, giving a richer positional signal than simple indices.

Random walk encodings: use the k -step random walk landing probabilities $\text{RWPE}(v)_i = (A^i)_{vv}$ (the probability of returning to v in exactly i steps). These are automatically invariant under graph automorphisms and capture both local and global structure.

Both types of positional encodings are sections of the fibration over **Grph** described in Chapter 14: they assign to each graph a positional embedding consistent with the graph's automorphism group, and the consistency condition is the naturality of the section with respect to graph isomorphisms.

Chapter 29

Transformers and the Category of Sequence-to-Sequence Maps

29.1 The Transformer as a Composite Morphism

The Transformer architecture (Vaswani et al., 2017) processes a sequence of tokens through alternating layers of self-attention and feed-forward networks, producing a new sequence of the same length at each layer. Its success across language, vision, audio, protein structure, and code suggests that it captures something fundamental about sequential relational data. The categorical framework makes this precise: the Transformer is a morphism in a specific monoidal category of sequence transformations, and its structural properties — equivariance, compositionality, universality — are consequences of the categorical axioms.

Define the category **Seq_d** whose objects are pairs (n, X) with $n \in \mathbb{N}$ (sequence length) and $X \in \mathbb{R}^{n \times d}$ (a matrix of n token embeddings of dimension d), and whose morphisms $(f, L) : (n, X) \rightarrow (m, Y)$ are pairs of a function $f : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ on positions and a linear map $L : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfying $Y_{f(i)} = L(X_i)$ for all i . A Transformer layer is a morphism in **Para(Seq_d)**: it takes an attributed sequence (n, X) and a parameter tuple $\theta = (W_Q, W_K, W_V, W_O, W_1, W_2, b_1, b_2, \gamma, \beta)$ and produces (n, Y) with the same length but updated embeddings.

The full Transformer with L layers is the composition of L such morphisms in **Para(Seq_d)**:

$$\text{Transformer}_\Theta = \text{Block}_{\theta_L} \circ \dots \circ \text{Block}_{\theta_1} : (n, X_0) \mapsto (n, X_L),$$

with total parameter space $\Theta = \Theta_1 \times \dots \times \Theta_L$. The composition is well-typed because each block preserves the sequence length n , making the output of one block a valid input to the next. The backpropagation functor $\text{BP} : \mathbf{Para}(\mathbf{Seq}_d) \rightarrow \mathbf{Lens}(\mathbf{Seq}_d)$ gives the gradient computation for the full Transformer by the chain rule (lens composition formula, Chapter 3).

29.2 Self-Attention as Profunctor Composition

The scaled dot-product attention mechanism computes:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \quad Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

where $W_Q, W_K \in \mathbb{R}^{d \times d_k}$ and $W_V \in \mathbb{R}^{d \times d_v}$. The attention weight matrix $A = \text{softmax}(QK^\top/\sqrt{d_k}) \in \mathbb{R}^{n \times n}$ is row-stochastic: each row sums to one, making A a Markov kernel on $\{1, \dots, n\}$.

As established in Chapter 15 (Ends and Coends), the attention output can be written as a coend:

$$\text{Attn}(Q, K, V)_i = \int^{j \in \mathbf{Pos}} A(i, j) \cdot V_j = \sum_j A_{ij} V_j,$$

where $\mathbf{Pos} = \{1, \dots, n\}$ is the discrete category of positions, $A : \mathbf{Pos} \nrightarrow \mathbf{Pos}$ is the attention profunctor, and $V : \mathbf{Pos} \rightarrow \mathbb{R}^{d_v}$ is the value functor. This profunctor perspective makes the attention mechanism a coend computation: it computes the Kan extension of the value assignment along the attention profunctor.

Attention is *permutation equivariant*: if we permute the input sequence by a bijection $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, then $\text{Attn}(Q_\sigma, K_\sigma, V_\sigma)_{\sigma(i)} = \text{Attn}(Q, K, V)_i$, where subscript σ denotes row permutation. This is naturality of the coend with respect to automorphisms of $\mathbf{Pos}$: relabelling the positions (a morphism in $\mathbf{Pos}$) commutes with the coend computation.

Multi-head attention with H heads is:

$$\text{MHA}(X) = \text{Concat}(h_1, \dots, h_H)W^O, \quad h_i = \text{Attn}(XW_i^Q, XW_i^K, XW_i^V).$$

Categorically, the H heads are H profunctors $A_1, \dots, A_H : \mathbf{Pos} \nrightarrow \mathbf{Pos}$, and multi-head attention is their parallel composition in $\mathbf{Prof}$ (the bicategory of profunctors, Chapter 15) followed by the output projection W^O . The monoidal structure of $\mathbf{Prof}$ guarantees that multi-head attention is a well-defined morphism, and the associativity and interchange laws guarantee that the order of head combination does not matter (up to W^O).

29.3 Positional Encodings and the Representation of Sequence Order

Self-attention is permutation equivariant by construction — it treats all positions identically. This is a feature (global context access) and a limitation (no intrinsic notion of order). Positional encodings add position-dependent information to break this symmetry.

The *sinusoidal positional encoding* of the original Transformer:

$$\text{PE}_{(i, 2k)} = \sin(i/10000^{2k/d}), \quad \text{PE}_{(i, 2k+1)} = \cos(i/10000^{2k/d}),$$

is a natural transformation $\text{PE} : \mathbf{Pos} \Rightarrow \mathbb{R}^d$ from the discrete position category to the embedding space. Each frequency $10000^{2k/d}$ gives a sinusoidal basis function for position

i , and the encoding has the property that $\text{PE}_{i+\delta}$ is a linear function of PE_i (rotation in the frequency domain), enabling the model to attend by relative offset.

Rotary Positional Encoding (RoPE; Su et al., 2024) encodes position by rotating query and key vectors: the query at position i is rotated by $R_{i\theta}$ (a block-diagonal rotation matrix), and the key at position j by $R_{j\theta}$, so the inner product $q_i^\top k_j = (R_{i\theta}q)^\top (R_{j\theta}k) = q^\top R_{(i-j)\theta}k$ depends only on the relative position $i - j$. The categorical structure is a group action of $(\mathbb{Z}, +)$ on $\mathbb{R}^{d_k}$ by rotations: the representation $\rho_\theta : n \mapsto R_{n\theta}$ is a group homomorphism $\mathbb{Z} \rightarrow O(d_k)$, and RoPE is the query-key inner product made $\mathbb{Z}$ -equivariant by this representation. The equivariance (relative-position dependence) is then a consequence of the general theory of G -equivariant morphisms (Chapter 18) applied to $G = (\mathbb{Z}, +)$.

29.4 The Feed-Forward Network as Memory and Universal Approximator

Each Transformer block contains a position-wise feed-forward network:

$$\text{FFN}(x) = W_2 \sigma(W_1 x + b_1) + b_2, \quad x \in \mathbb{R}^d,$$

applied independently to each position: $\text{FFN}(X)_i = \text{FFN}(X_i)$. The FFN is a morphism in **Para(Seq_d)** that factors through the projection to individual embeddings, applies the two-layer MLP to each, and reassembles.

The *key-value memory* interpretation (Geva et al., 2021): the first layer W_1 acts as a set of keys, activating when the input matches. The nonlinearity σ thresholds the activation. The second layer W_2 stores the corresponding values, read out by the activated keys. Formally, the FFN computes:

$$\text{FFN}(x) = \sum_{i=1}^{4d} \sigma(w_1^{(i)} \cdot x + b_1^{(i)}) \cdot w_2^{(i)},$$

where $w_1^{(i)}$ is the i -th row of W_1 (a key) and $w_2^{(i)}$ is the i -th column of W_2 (a value). This is a soft key-value lookup: the activation $\sigma(w_1^{(i)} \cdot x)$ measures the match between x and key $w_1^{(i)}$, and the output is the weighted sum of the corresponding values.

The universal approximation theorem for FFNs (Hornik, 1991) states that for any continuous $f : K \rightarrow \mathbb{R}^d$ on compact $K \subseteq \mathbb{R}^d$ and any $\varepsilon > 0$, there exist W_1, W_2, b_1, b_2 with $\|W_2 \sigma(W_1 \cdot + b_1) - f\|_{C(K)} < \varepsilon$. In categorical terms (Chapter 12): the image of the Para morphism $(\Theta, \text{FFN}_\theta) \in \mathbf{Para}(\mathbf{Smooth})$ is dense in $C(K, \mathbb{R}^d)$ with the uniform topology — the Kan extension of any continuous function from training data can be approximated to arbitrary accuracy by a two-layer FFN.

29.5 Residual Connections and Layer Normalisation in the Transformer Block

The full Transformer block (Pre-LN variant) is:

$$\begin{aligned} X' &= X + \text{MHA}(\text{LN}(X)), \\ X'' &= X' + \text{FFN}(\text{LN}(X')). \end{aligned}$$

In categorical terms, each component is a morphism in $\mathbf{Para}(\mathbf{Seq}_d)$:

- LN: the quotient morphism of Chapter 24, a morphism in the subcategory of unit-sphere embeddings.
- MHA: the profunctor composition of Chapter 29, a morphism in $\mathbf{Para}(\mathbf{Seq}_d)$.
- $X + \text{MHA}(\text{LN}(X))$: the residual codiagonal of Chapter 24, using the bimonoid structure (Δ, ∇) on $\mathbb{R}^d$.

The block is thus a composite of three categorical operations, each with a clean categorical interpretation.

The backward pass through a Transformer block is the lens composition of the backward passes through each component. The residual connections ensure that the gradient passes through the block intact: even if MHA or FFN has near-zero gradient contribution, the direct path $\delta X \mapsto \delta X$ through the skip connection guarantees that gradients reach earlier layers.

29.6 Attention as a Kernel Method and Linearisation

The attention weight $A_{ij} = \text{softmax}(q_i \cdot k_j / \sqrt{d_k})$ can be written as:

$$A_{ij} = \frac{K(q_i, k_j)}{\sum_{j'} K(q_i, k_{j'})}, \quad K(q, k) = \exp(q \cdot k / \sqrt{d_k}),$$

where K is the (unnormalised) exponential kernel. The attention mechanism is therefore a kernel machine: it computes a weighted average of value vectors with weights given by softmax-normalised kernel evaluations. In the RKHS perspective (Chapter 12), K defines a Hilbert space $\mathcal{H}_K$ via the Moore-Aronszajn theorem, and the attention output is the kernel mean embedding evaluated at the value vectors.

Linearised attention (Katharopoulos et al., 2020) approximates $K(q, k) \approx \phi(q)^\top \phi(k)$ with a low-rank feature map $\phi : \mathbb{R}^{d_k} \rightarrow \mathbb{R}^m$ (e.g., random Fourier features for the Gaussian kernel). The linearised attention is:

$$\text{LinAttn}(Q, K, V)_i = \frac{\phi(q_i)^\top (\sum_j \phi(k_j) v_j^\top)}{\phi(q_i)^\top (\sum_j \phi(k_j))},$$

computable in $O(nm)$ time (the sums $\sum_j \phi(k_j) v_j^\top$ and $\sum_j \phi(k_j)$ can be precomputed). Categorically, the linearisation replaces the full rank- n profunctor $A : \mathbf{Pos} \nrightarrow \mathbf{Pos}$ with

a rank- m factored profunctor $A \approx \Phi_Q \Phi_K^\top$, where $\Phi_Q, \Phi_K \in \mathbb{R}^{n \times m}$ are the query and key feature matrices. The expressivity loss is the loss of the full-rank profunctor structure; the computational gain is from $O(n^2)$ to $O(nm)$.

29.7 Turing Completeness and the Categorical Universality of Transformers

Transformers (with appropriate positional encodings and sufficient depth/width) are Turing complete: they can simulate any Turing machine computation on inputs of bounded length, given sufficient precision and context length (Pérez et al., 2019). The categorical account uses the theory of universal objects in cartesian closed categories.

A *cartesian closed category* (CCC) has, for each pair (A, B) , an *exponential object* $[A \rightarrow B]$ representing all morphisms from A to B , with the evaluation morphism $\text{ev} : [A \rightarrow B] \times A \rightarrow B$ and currying isomorphism $\mathcal{C}(C \times A, B) \cong \mathcal{C}(C, [A \rightarrow B])$. The Lambek-Scott theorem (Chapter 33) identifies CCCs as the categorical semantics of the simply typed lambda calculus, a Turing-complete computational model.

The Transformer’s attention mechanism implements a form of differentiable memory access: given a query, it retrieves a weighted combination of stored key-value pairs. This is the categorical ev morphism applied softly: rather than retrieving a single stored value (the sharp ev), attention retrieves a weighted combination (a soft ev). In the limit of sharp attention ($A_{ij} \rightarrow \delta_{j, j^*(i)}$ for the argmax position $j^*(i)$), the Transformer implements exact key-value lookup, and the attention heads can be configured to implement arbitrary finite-state computation.

Turing completeness then follows: any Turing machine transition function (a finite state machine reading and writing a tape) can be implemented by a single Transformer layer that (1) reads the current state from a designated position in the sequence (via attention), (2) computes the new state and tape symbol (via the FFN), and (3) writes the result to the output sequence. Sufficient layers allow simulation of any bounded computation.

The categorical content: the Transformer is a morphism in $\mathbf{Para}([\mathbf{Pos}^{\text{op}}, \mathbf{Set}])$, the Para construction over the presheaf category on positions (a CCC). The Turing completeness result says that the image of this Para morphism is dense in the space of all computable sequence-to-sequence functions, which is the categorical universal approximation theorem for Transformers.

Chapter 30

Reinforcement Learning, Markov Decision Processes, and Categorical Dynamics

30.1 Markov Decision Processes as Polynomial Coalgebras

Reinforcement learning (RL) is the study of agents that learn to act in an uncertain environment by trial and error. The mathematical framework is the Markov decision process (MDP), which models the environment as a stateful system that responds to actions with stochastic transitions and numerical rewards. The categorical account identifies MDPs with specific morphisms in **BorelStoch** and the RL problem with a fixed-point equation in the Markov category.

An MDP is a tuple (S, A, P, R, γ) : a state space S , action space A , transition kernel $P : S \times A \rightarrow \mathcal{G}(S)$, reward function $R : S \times A \rightarrow \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. In the Markov category **BorelStoch** (Chapter 6), the transition P is a Markov kernel: a morphism $P : S \times A \rightarrow S$ in **BorelStoch**. A policy $\pi : S \rightarrow \mathcal{G}(A)$ is also a Markov kernel: a morphism $\pi : S \rightarrow A$ in **BorelStoch**.

As established in Chapter 5, an MDP is a *polynomial coalgebra*. Define the polynomial functor $p(X) = \mathcal{G}(S) \times (A \rightarrow X)$ (a distribution over next states, together with a continuation for each action). An MDP coalgebra is a pair (S, c) where $c : S \rightarrow p(S) = \mathcal{G}(S) \times A^S$ maps each state to a distribution over reward-weighted next states and a function from actions to next states. The transition kernel P and reward R together specify c .

The *policy-transition composite* is the Kleisli composition (Chapter 13) of π and P in the Girmonad:

$$P^\pi(s' \mid s) = \int_A P(s' \mid s, a) \pi(a \mid s) da, \quad P^\pi : S \rightarrow S \text{ in } \mathbf{BorelStoch}.$$

This is the “closed-loop” transition kernel: the Markov chain on S obtained by following policy π . The stationary distribution μ^π of this chain satisfies $(P^\pi)_* \mu^\pi = \mu^\pi$ (a fixed-point

condition in the Giry monad), and ergodicity of the MDP implies unique existence of μ^π .

30.2 The Bellman Equation as a Fixed-Point in the Giry Monad

The value function $V^\pi : S \rightarrow \mathbb{R}$ of a policy π is defined as the expected discounted return:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right].$$

It satisfies the *Bellman equation*:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} [V^\pi(s')]] = (\mathcal{T}^\pi V^\pi)(s),$$

where $\mathcal{T}^\pi : (S \rightarrow \mathbb{R}) \rightarrow (S \rightarrow \mathbb{R})$ is the *Bellman operator*. Decomposing:

$$(\mathcal{T}^\pi V)(s) = R^\pi(s) + \gamma(P^\pi)_* V(s), \quad R^\pi(s) = \mathbb{E}_{a \sim \pi} [R(s, a)],$$

where $(P^\pi)_* V(s) = \int_S V(s') P^\pi(s' | s) ds'$ is the expected value of V under the transition kernel. The operator $\mathcal{T}^\pi$ is a γ -contraction in the L^∞ norm: $\|\mathcal{T}^\pi V - \mathcal{T}^\pi V'\|_\infty \leq \gamma \|V - V'\|_\infty$. By the Banach fixed-point theorem in the Lawvere metric space $(C_b(S), \|\cdot\|_\infty)$ (Chapter 7), $\mathcal{T}^\pi$ has a unique fixed point V^π , and value iteration $V_{k+1} = \mathcal{T}^\pi V_k$ converges geometrically: $\|V_k - V^\pi\|_\infty \leq \gamma^k \|V_0 - V^\pi\|_\infty$.

Theorem 30.1 (Bellman Optimality). *Define the optimal Bellman operator $(\mathcal{T}^* V)(s) = \sup_{a \in A} [R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} [V(s')]]$. Then:*

1. $\mathcal{T}^*$ is a γ -contraction on $(C_b(S), \|\cdot\|_\infty)$.
2. Its unique fixed point $V^* : S \rightarrow \mathbb{R}$ satisfies $V^*(s) = \sup_\pi V^\pi(s)$ for all s .
3. The optimal policy $\pi^*(a | s) = \delta_{a^*(s)}$, where $a^*(s) = \arg \max_a [R(s, a) + \gamma \mathbb{E}_{s'} [V^*(s')]]$, achieves $V^{\pi^*} = V^*$.

Categorically, V^ is the terminal object in the category of solutions to the Bellman optimality equation, ordered by pointwise comparison: every other fixed point V of $\mathcal{T}^*$ satisfies $V \leq V^*$ pointwise.*

Proof of contraction. $|(\mathcal{T}^* V)(s) - (\mathcal{T}^* V')(s)| = |\sup_a [\gamma \mathbb{E}_{s'} [V(s') - V'(s')]]| \leq \gamma \|V - V'\|_\infty$ by the monotonicity of sup and the linearity of expectation. $\square$

30.3 Deep Q-Learning and Parametric Bellman Residuals

Deep Q-learning (DQN; Mnih et al., 2015) approximates the optimal action-value function $Q^* : S \times A \rightarrow \mathbb{R}$ (where $Q^*(s, a) = R(s, a) + \gamma \mathbb{E}_{s'} [\max_{a'} Q^*(s', a')]$) by a neural network $Q_\theta : S \times A \rightarrow \mathbb{R}$ in **Para(Smooth)**. The training objective is the Bellman residual:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s, a, r, s') \sim \mathcal{D}} \left[(Q_\theta(s, a) - (r + \gamma \max_{a'} Q_\theta(s', a')))^2 \right],$$

where $\mathcal{D}$ is the replay buffer and θ^- are the *target network* parameters (a periodically updated copy of θ).

The target network is a categorical construction: it replaces the self-referential fixed-point equation $Q_\theta = \mathcal{T}^*(Q_\theta)$ (which would require differentiating through Q_θ on both sides simultaneously) with a two-step iteration. In the first step, the target $y = r + \gamma \max_{a'} Q_{\theta^-}(s', a')$ is computed with θ^- held fixed — a morphism in **BorelStoch** (stochastic, because s' is sampled). In the second step, θ is updated by gradient descent on the squared error — a morphism in **Para(Smooth)**. The target network θ^- is periodically synchronised with θ , making the two-step iteration converge to the Bellman fixed point.

Categorically, this is the *splitting of a fixed-point equation*: the self-referential morphism $Q_\theta = \mathcal{T}^*(Q_\theta)$ is factored as a pair of morphisms ($f : \theta \mapsto Q_\theta$, $g : Q_\theta \mapsto \mathcal{T}^*(Q_\theta)$) in a fibred category over the parameter space, and the iteration alternates between these two factors. This is the categorical analogue of the EM algorithm (Chapter 19): each step improves one factor while holding the other fixed.

30.4 Policy Gradient Methods and the Natural Policy Gradient

Policy gradient methods directly parametrise the policy $\pi_\theta : S \rightarrow \mathcal{G}(A)$ and optimise the expected return $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t \gamma^t R_t]$. The policy gradient theorem (Sutton et al., 1999) gives:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a | s) \cdot Q^\pi(s, a)],$$

where $\rho^\pi(s) = \sum_{t=0}^\infty \gamma^t P(s_t = s)$ is the discounted state visitation measure under π_θ . The gradient $\nabla_\theta \log \pi_\theta(a | s)$ is the *score function* of the policy (the score natural transformation of Chapter 23 applied to π_θ), and the theorem says that the policy gradient equals the expected score times the Q-value.

The backpropagation functor $\text{BP} : \mathbf{Para}(\mathbf{BorelStoch}) \rightarrow \mathbf{Lens}(\mathbf{BorelStoch})$ applied to the policy $\pi_\theta : S \rightarrow \mathcal{G}(A)$ gives a lens whose backward component is $\nabla_\theta \log \pi_\theta(a | s)$ (the derivative of the log-policy). The policy gradient theorem then says that the gradient of J is the expectation of the lens's backward component weighted by the Q-value — a morphism in **BorelStoch** applied to the lens output.

The *natural policy gradient* (Kakade, 2002) replaces $\nabla_\theta J$ with the Fisher-natural gradient:

$$\tilde{\nabla}_\theta J = F_\theta^{-1} \nabla_\theta J, \quad F_\theta = \mathbb{E}_{s,a}[\nabla_\theta \log \pi_\theta(a | s) \nabla_\theta \log \pi_\theta(a | s)^\top],$$

where F_θ is the Fisher information matrix of the policy family $\{\pi_\theta\}_\theta$ (Chapter 19). The natural policy gradient is invariant under reparametrisation of θ (naturality of the musical isomorphism $F_\theta^{-1} : T^*\Theta \rightarrow T\Theta$) and converges to the optimal policy faster than the standard policy gradient in practice.

30.5 Actor-Critic as a Bidirectional Morphism

Actor-critic methods (Konda-Tsitsiklis, 2000) maintain two networks simultaneously:

- *Actor* $\pi_\theta : S \rightarrow \mathcal{G}(A)$: the policy, updated by policy gradient.
- *Critic* $V_\phi : S \rightarrow \mathbb{R}$: an approximation to the value function, updated by Bellman residual.

The advantage function $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ is estimated as $\hat{A} = r + \gamma V_\phi(s') - V_\phi(s)$ (the *temporal difference error*), and the policy gradient is approximated as:

$$\hat{\nabla}_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a | s) \cdot \hat{A}.$$

The categorical structure is a *bidirectional computation*: a lens $\ell : \mathbf{Para}(\mathbf{BorelStoch})$:

- Forward pass (**get**): actor computes $a \sim \pi_\theta(\cdot | s)$, environment returns r, s' , critic computes $V_\phi(s)$ and $V_\phi(s')$.
- Backward pass (**put**): actor receives $\hat{A} = r + \gamma V_\phi(s') - V_\phi(s)$ and updates θ by policy gradient; critic updates ϕ to reduce the Bellman residual $(V_\phi(s) - (r + \gamma V_\phi(s')))^2$.

The actor-critic lens is thus the composition of the actor lens (computing actions and receiving advantage signals) with the critic lens (computing value estimates and receiving Bellman residual signals). The lens composition formula (Chapter 3) ensures that the backward passes are computed in the correct order.

30.6 Reinforcement Learning from Human Feedback

Reinforcement Learning from Human Feedback (RLHF; Christiano et al., 2017) replaces the MDP reward R with a reward learned from human preference comparisons. A reward model $R_\phi : S \times A \rightarrow \mathbb{R}$ is trained by the Bradley-Terry model:

$$P(\text{response } y_1 \succ y_2 | x) = \sigma(R_\phi(x, y_1) - R_\phi(x, y_2)),$$

where σ is the logistic function and $y_1 \succ y_2$ means a human prefers y_1 over y_2 given context x . The policy is then optimised against the learned reward, with a KL constraint preventing deviation from a reference policy π_{ref} :

$$\max_{\pi} \mathbb{E}_{a \sim \pi} [R_\phi(s, a)] - \beta \text{KL}(\pi \| \pi_{\text{ref}}).$$

The optimal policy for this objective has a closed-form solution:

$$\pi^*(a | s) = \frac{\pi_{\text{ref}}(a | s) \exp(R_\phi(s, a)/\beta)}{Z(s)}, \quad Z(s) = \sum_{a'} \pi_{\text{ref}}(a' | s) \exp(R_\phi(s, a')/\beta).$$

This is the *Gibbs posterior* (Chapter 22) with the reward model as the energy and the reference policy as the prior. The RLHF training procedure approximates this Gibbs posterior within the parametric class of neural network policies.

The categorical structure: π^* is the fixed point of the Bayesian update morphism $\pi \mapsto \pi_{\text{ref}} \cdot e^{R_\phi/\beta}/Z$ in **BorelStoch**, which is exactly the Kleisli morphism of the Giry monad applied to the exponential family (Chapter 13). The KL constraint is the morphism $\text{KL}(\pi \parallel \pi_{\text{ref}}) = \mathbb{E}_\pi[\log \pi / \pi_{\text{ref}}]$ in **BorelStoch** (the log-likelihood ratio, a morphism in **BorelStoch** as shown in Chapter 22), and minimising this constraint is equivalent to maximising the normalisation constant $Z(s)$ — the evidence in the Bayesian interpretation.

Direct Preference Optimisation (DPO; Rafailov et al., 2023) bypasses the explicit reward model by directly parametrising the optimal policy via the implicit reward $R_\phi(s, a) = \beta \log(\pi_\theta(a \mid s) / \pi_{\text{ref}}(a \mid s)) + \beta \log Z(s)$, and optimising the policy directly on human preference data. The DPO loss is:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right].$$

Categorically, DPO replaces the composition of two morphisms in **Para(BorelStoch)** (learn reward model, then train policy) with a single morphism (train policy directly on preferences), reducing the variance of the gradient estimate by eliminating the intermediate reward model.

Chapter 31

Neural Scaling Laws and the Category of Model Families

31.1 Empirical Scaling Laws and Power-Law Behaviour

Among the most striking empirical discoveries in modern deep learning is the near-perfect power-law relationship between model performance and scale. Kaplan et al. (2020) observed that for language models trained on internet text, the cross-entropy loss decreases as:

$$\mathcal{L}(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \mathcal{L}(D) \approx \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \mathcal{L}(C) \approx \left(\frac{C_c}{C}\right)^{\alpha_C},$$

with exponents $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $\alpha_C \approx 0.050$ across many orders of magnitude in each variable. The constants N_c, D_c, C_c are dataset-specific thresholds.

These power laws hold over at least six orders of magnitude in model size and training compute, suggesting that they reflect fundamental mathematical structure rather than coincidental regularities. The Chinchilla scaling laws (Hoffmann et al., 2022) refine the picture: for a fixed compute budget $C \approx 6ND$ (the FLOPs required to train an N -parameter model on D tokens, using ≈ 6 multiply-accumulate operations per parameter per token), the optimal allocation minimises:

$$\mathcal{L}(N, D) \approx E + \frac{A}{N^\alpha} + \frac{B}{D^\beta},$$

where E is the irreducible entropy of the data (the best possible loss), A/N^α is the loss from insufficient model capacity, and B/D^β is the loss from insufficient data. Minimising over N subject to $6ND = C$ gives optimal $N^* \propto C^{1/2}$ and $D^* \propto C^{1/2}$ (equal scaling of parameters and data).

31.2 The Moduli Space of Parametric Model Families

To give the scaling laws a categorical foundation, we construct the *moduli space* of neural network model families. A *model family* is an object in the Para construction: a pair

(Θ_N, f_N) where Θ_N is the N -dimensional parameter space (a smooth manifold) and $f_N : \Theta_N \times X \rightarrow Y$ is the implementation function in **Para(Smooth)**.

Define the category **ModFam** of model families:

- *Objects*: pairs (Θ_N, f_N) with $N = \dim \Theta_N$.
- *Morphisms*: $(\Theta_N, f_N) \rightarrow (\Theta_M, f_M)$ is a smooth map $\phi : \Theta_N \rightarrow \Theta_M$ (a *model compression or distillation map*) satisfying $f_M \circ (\phi \times \text{id}_X) = f_N$ on the training distribution.
- *Equivalences*: reparametrisations $\phi : \Theta_N \xrightarrow{\sim} \Theta_N$ with $f_N \circ (\phi \times \text{id}_X) = f_N$ (isomorphisms in **ModFam**).

The *moduli space* $\mathcal{M}$ of model families is the stack of equivalence classes of **ModFam** under the isomorphism relation. Two model families are *equivalent* if they compute the same function on the data distribution up to reparametrisation, and $\mathcal{M}$ parametrises all intrinsically distinct model families.

The scaling law $\mathcal{L}(N) = \min_{\theta \in \Theta_N} \mathcal{L}(\theta)$ is a function on the moduli space: it assigns to each equivalence class of N -parameter families its minimum achievable loss. As a function $\mathcal{L} : \mathcal{M} \rightarrow \mathbb{R}$, it is the *Bayes risk* of the model family (the best achievable loss within the parametric class), and the power-law behaviour $\mathcal{L}(N) \sim N^{-\alpha}$ is the statement that the Bayes risk decreases as a power of the model complexity.

31.3 Metric Entropy, Covering Numbers, and the Scaling Exponent

The scaling exponent α is determined by the *metric entropy* of the model family in the loss metric. The covering number $\mathcal{N}(\mathcal{F}, \varepsilon, d)$ is the minimum number of ε -balls (in metric d) needed to cover the function class $\mathcal{F} = \{f_\theta : \theta \in \Theta_N\}$. The logarithm $H(\mathcal{F}, \varepsilon, d) = \log \mathcal{N}(\mathcal{F}, \varepsilon, d)$ is the *metric entropy*.

For parametric classes with N parameters, the metric entropy scales as $H(\mathcal{F}_N, \varepsilon, d) \approx N \log(1/\varepsilon)$ (by a standard counting argument: there are roughly $(1/\varepsilon)^N$ distinct ε -grids in Θ_N). The Rademacher complexity bound gives:

$$\mathcal{L}^*(m) \lesssim \mathcal{L}(N) + \sqrt{\frac{H(\mathcal{F}_N, 1/\sqrt{m}, d)}{m}} \approx \mathcal{L}(N) + \sqrt{\frac{N \log m}{m}},$$

where m is the training set size. Optimising over N gives $N^* \propto m/\log m$, and the optimal loss scales as $\mathcal{L}^*(m) \sim (\log m/m)^\alpha$ for some α determined by the smoothness of the target function.

In the enriched category framework (Chapter 7), the covering number is a morphism in the Lawvere metric category of function classes: $\mathcal{N}(\mathcal{F}_N, \varepsilon, d)$ is the hom-object from $\mathcal{F}_N$ to the ε -covering in the metric $(C(X), d)$. The scaling law exponent α is the slope of the log-log plot of the hom-object versus the dimension N of the parameter space.

Theorem 31.1 (Scaling Law Exponent from Smoothness; informal). *Let $f^* : X \rightarrow Y$ be a target function in the Sobolev space $W^{s,2}(X)$ (functions with s square-integrable derivatives). Let $\mathcal{F}_N$ be the class of two-layer neural networks with N hidden units and activation in C^s . Then:*

$$\mathcal{L}^*(N) = \inf_{\theta} \|f_{\theta} - f^*\|_{L^2}^2 \lesssim N^{-2s/d},$$

giving scaling exponent $\alpha = 2s/d$ (the approximation rate is determined by the smoothness s relative to the input dimension d). For language models, d is effectively the dimension of the token embedding space and s captures the smoothness of the language distribution; the empirical $\alpha \approx 0.076$ corresponds to $s/d \approx 0.038$.

The categorical content: the Sobolev space $W^{s,2}(X)$ is the RKHS of the Sobolev kernel $K_s(x, y) = \sum_{|\alpha| \leq s} D^{\alpha} \delta_x \cdot D^{\alpha} \delta_y$ (the inner product of all partial derivatives up to order s), and the approximation rate $N^{-2s/d}$ is the rate at which the Yoneda representation of f^* in this RKHS is approximated by the Yoneda representation of f_{θ} (Chapter 8). The scaling exponent is thus a property of the Yoneda embedding of the target function in the RKHS.

31.4 Phase Transitions and Emergent Capabilities

A striking feature of neural scaling laws is the occurrence of *phase transitions*: abrupt changes in qualitative capability at specific scale thresholds. Wei et al. (2022) document numerous such transitions: the ability to perform few-shot arithmetic, multi-step reasoning, and code generation emerge suddenly at critical model sizes, with performance near chance below the threshold and rapidly improving above it.

The categorical account uses Morse theory on the loss landscape. The loss landscape $\mathcal{L} : \Theta_N \rightarrow \mathbb{R}$ changes qualitatively as N increases: new critical points appear, their Morse indices change, and the topology of sublevel sets $\{\theta : \mathcal{L}(\theta) \leq c\}$ changes at bifurcation values of N .

Define the *capability region* for task τ as the sublevel set:

$$\mathcal{C}_{\tau}(N) = \{\theta \in \Theta_N : \mathcal{L}_{\tau}(\theta) \leq \varepsilon_{\tau}\} \subseteq \Theta_N,$$

where $\mathcal{L}_{\tau}$ is the loss on task τ and ε_{τ} is the threshold for “capable performance”. An emergent capability corresponds to the appearance of $\mathcal{C}_{\tau}(N)$ as a non-empty connected set as N crosses a critical threshold N_{τ}^* : for $N < N_{\tau}^*$, $\mathcal{C}_{\tau}(N) = \emptyset$; for $N > N_{\tau}^*$, $\mathcal{C}_{\tau}(N)$ is non-empty.

The phase transition at N_{τ}^* is a *categorical bifurcation*: the topology of the fibred category $\mathcal{M}_{\tau} = \{(\Theta_N, \mathcal{C}_{\tau}(N)) : N \in \mathbb{N}\}$ (a sub-fibration of the moduli space fibred over $\mathbb{N}$) changes at $N = N_{\tau}^*$. The emergence of capability is the appearance of a new connected component in the fibre $\mathcal{C}_{\tau}(N)$, which is a categorical change of structure that corresponds to a change in the Betti numbers of the sublevel set.

31.5 The Chinchilla Optimum as Categorical Optimal Transport

The Chinchilla optimal allocation of compute between parameters and data can be given a categorical optimal transport interpretation. The compute budget $C = 6ND$ is a resource constraint, and the problem is to allocate C between parameters (N) and data (D) to minimise the loss $\mathcal{L}(N, D) = E + A/N^\alpha + B/D^\beta$.

Define the *compute category* **Comp** with objects (N, D) (allocation of compute between parameters and data) and morphisms $(N, D) \rightarrow (N', D')$ whenever $N'D' \leq ND$ (more efficient allocations dominate less efficient ones). The loss $\mathcal{L} : \mathbf{Comp} \rightarrow \mathbb{R}$ is a decreasing functor (better allocations give lower loss), and the Chinchilla optimal allocation $\text{OPT}(C) = (N^*(C), D^*(C))$ is the object in **Comp** that minimises $\mathcal{L}$ subject to $6ND = C$ — the terminal object in the subcategory of C -constrained allocations.

In the Wasserstein optimal transport framework (Chapter 23), the compute budget C is a resource to be optimally transported between two “sinks”: the parameter sink (which reduces A/N^α) and the data sink (which reduces B/D^β). The Chinchilla optimal allocation is the solution to the optimal transport problem of distributing C between these two sinks with cost functions $N \mapsto A/N^\alpha$ and $D \mapsto B/D^\beta$, subject to the constraint $6ND = C$. The power-law form of the cost functions means that the optimal transport map is itself a power law in C , giving $N^* \propto C^a$ and $D^* \propto C^b$ for constants a, b determined by α and β .

31.6 PAC-Bayes and the Information-Theoretic Lower Bound

The PAC-Bayes framework (Chapter 22) provides an information-theoretic lower bound on the scaling law exponent. For a model family $\mathcal{F}_N = \{f_\theta : \theta \in \Theta_N\}$ with a Gaussian prior $P = \mathcal{N}(0, \sigma^2 I)$ on Θ_N , the PAC-Bayes bound gives:

$$\mathcal{L}(N, m) \leq \hat{\mathcal{L}}(N, m) + \sqrt{\frac{\text{KL}(Q \| P) + \log(m/\delta)}{2m}},$$

where Q is the posterior concentrated near the optimum θ^* and m is the training set size. For the Gaussian posterior $Q = \mathcal{N}(\theta^*, \sigma_Q^2 I)$ with optimal $\sigma_Q^2 \propto 1/m$, the KL divergence is:

$$\text{KL}(Q \| P) = \frac{N}{2} \left(\frac{\sigma_Q^2}{\sigma^2} - 1 - \log \frac{\sigma_Q^2}{\sigma^2} + \frac{\|\theta^*\|^2}{N\sigma^2} \right) \approx \frac{N \log m}{2m},$$

giving the bound $\mathcal{L}(N, m) \lesssim \mathcal{L}^*(N) + \sqrt{N \log m / m}$. Optimising over N gives $N^* \propto \sqrt{m / \log m}$ and the scaling law $\mathcal{L}^*(m) \sim (m / \log m)^{-\alpha}$.

Categorically, the PAC-Bayes bound is a natural inequality between two functors on **BorelStoch**: the “true loss” functor $\mathcal{L} : \mathcal{G}(\Theta_N) \rightarrow \mathbb{R}$ and the “empirical loss plus complexity penalty” functor $\hat{\mathcal{L}} + \text{complexity} : \mathcal{G}(\Theta_N) \times X^m \rightarrow \mathbb{R}$. The bound is the statement that the first functor is dominated by the second (up to a high-probability event over the data draw), which is a lax natural transformation in **BorelStoch**. The scaling law exponent α is the rate of convergence of this lax natural transformation, determined by the metric

entropy of the model family.

Chapter 32

Categorical Linear Algebra, Tensor Networks, and Quantum-Inspired Methods

32.1 Compact Closure, Traces, and the Partial Trace

The category $\mathbf{FVect}_{\mathbb{R}}$ is compact closed (Chapter 2): every object V has a dual $V^* = \text{Hom}_{\mathbb{R}}(V, \mathbb{R})$, and the snake equations give a canonical notion of transposition and trace. We now develop these structures more fully, with attention to their role in tensor network methods and the quantum-classical interface.

The *categorical trace* of an endomorphism $f : V \rightarrow V$ is:

$$\text{tr}(f) = \varepsilon_V \circ (\text{id}_{V^*} \otimes f) \circ \eta_V : \mathbb{R} \rightarrow \mathbb{R} \cong \mathbb{R},$$

where $\eta_V : \mathbb{R} \rightarrow V^* \otimes V$ sends $1 \mapsto \sum_i e_i^* \otimes e_i$ (for any basis $\{e_i\}$ of V) and $\varepsilon_V : V \otimes V^* \rightarrow \mathbb{R}$ sends $v \otimes f \mapsto f(v)$. Since a morphism $\mathbb{R} \rightarrow \mathbb{R}$ in $\mathbf{FVect}_{\mathbb{R}}$ is multiplication by a scalar, $\text{tr}(f) \in \mathbb{R}$ is the scalar obtained from this composition:

$$\text{tr}(f) = \sum_i e_i^*(f(e_i)) = \sum_i f_{ii}.$$

This is basis-independent (the Joyal-Street coherence theorem guarantees that any two basis choices give the same scalar) and is the usual matrix trace.

The *partial trace* over a subsystem W of an endomorphism $f : V \otimes W \rightarrow V \otimes W$ is:

$$\text{tr}_W(f) = (\text{id}_V \otimes \varepsilon_W) \circ (\text{id}_{V \otimes W^*} \otimes f) \circ (\text{id}_V \otimes \eta_W) : V \rightarrow V.$$

In matrix components: if $f_{ij,kl}$ (with i, k indexing V and j, l indexing W), then $\text{tr}_W(f)_{ik} = \sum_j f_{ij,kj}$. The partial trace is natural in V and W (a natural transformation between endomorphism functors), expressing basis-independence.

Proposition 32.1 (Naturality of the Partial Trace). *For any linear isomorphisms $g :$*

$V \xrightarrow{\sim} V'$ and $h : W \xrightarrow{\sim} W'$:

$$\mathrm{tr}_{W'}((g \otimes h) \circ f \circ (g^{-1} \otimes h^{-1})) = g \circ \mathrm{tr}_W(f) \circ g^{-1}.$$

This is the categorical statement that the partial trace is invariant under change of basis on the traced-over subsystem.

Proof. Expand $(g \otimes h) \circ f \circ (g^{-1} \otimes h^{-1})$ in the partial trace formula; the h and h^{-1} factors cancel in the summation over the basis of W' , leaving $g \circ \mathrm{tr}_W(f) \circ g^{-1}$. $\square$

Applications in machine learning: the Gram matrix $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$ (used in kernel methods) is the partial trace of the tensor $\phi(x_i) \otimes \phi(x_j)^*$ over the RKHS $\mathcal{H}$. The Fisher information matrix $\mathcal{I}(\theta)_{ij} = \mathbb{E}[(\partial_i \log p_{\theta})(\partial_j \log p_{\theta})]$ is the partial trace of the outer product $\nabla \log p_{\theta} \otimes (\nabla \log p_{\theta})^*$ over the data distribution. Both are categorical traces in the appropriate compact closed category.

32.2 Tensor Networks as String Diagrams in $\mathbf{FVect}_{\mathbb{R}}$

A *tensor network* is a collection of tensors (elements of tensor products of vector spaces) connected by contractions (summations over paired indices). By the Joyal-Street theorem (Chapter 2), every tensor network is a string diagram in the compact closed category $\mathbf{FVect}_{\mathbb{R}}$: boxes are tensors, wires are indices, and contractions are connections between output and input wires.

The *matrix product state* (MPS), also called a *tensor train*, represents a high-order tensor $T \in (\mathbb{R}^d)^{\otimes n}$ as:

$$T_{i_1 i_2 \dots i_n} = \sum_{\alpha_1, \dots, \alpha_{n-1}} A_{i_1, \alpha_1}^{[1]} A_{\alpha_1, i_2, \alpha_2}^{[2]} \cdots A_{\alpha_{n-1}, i_n}^{[n]},$$

where $A^{[k]} \in \mathbb{R}^{\chi_{k-1} \times d \times \chi_k}$ are *bond tensors* and χ_k is the *bond dimension* at position k . The string diagram is a linear chain: bond tensors are boxes with one physical index (pointing outward) and two bond indices (connecting adjacent boxes). Contracting adjacent bond indices gives the value of T at any fixed index tuple.

The MPS is a morphism in $\mathbf{FVect}_{\mathbb{R}}$: it is the composition of the n bond tensors along the chain, with physical indices as free outputs and bond indices contracted. The categorical content is that the MPS decomposes a high-dimensional morphism (the d^n -dimensional tensor T) into a composition of low-dimensional morphisms (the $\chi^2 d$ -dimensional bond tensors), reducing the parameter count from d^n to $n\chi^2 d$ — exponential to linear in n when χ is fixed.

Theorem 32.2 (MPS Approximation, via SVD). *Any tensor $T \in (\mathbb{R}^d)^{\otimes n}$ can be written exactly as an MPS with bond dimension $\chi \leq d^{\lfloor n/2 \rfloor}$ (by sequential SVD of bipartite reshaped matrices). For any $\chi < d^{\lfloor n/2 \rfloor}$, the best rank- χ MPS approximation is obtained by truncating the SVDs to the top χ singular values, and the approximation error is $\|T - T_{\chi}\|_F = \sqrt{\sum_k \sum_{\alpha > \chi} \sigma_{k,\alpha}^2}$ (the square root of the sum of discarded squared singular*

values). Categorically, the truncated MPS is the best rank- χ morphism in $\mathbf{FVect}_{\mathbb{R}}$ (with the Frobenius norm as the enriched hom) that approximates T .

In machine learning, the tensor train decomposition (Oseledets, 2011) compresses large weight tensors. A fully connected layer $W \in \mathbb{R}^{d^n \times d^n}$ is reshaped and approximated as an MPS with bond dimension χ , reducing the parameter count from d^{2n} to $O(n\chi^2 d^2)$. For $n = 10$, $d = 8$, $\chi = 8$, this reduces $\approx 10^{18}$ parameters to $\approx 5 \times 10^4$ — a reduction by 13 orders of magnitude, with controllable approximation error.

32.3 Tucker Decomposition and Convolutional Architecture

The *Tucker decomposition* represents a tensor $T \in \mathbb{R}^{d_1 \times \dots \times d_n}$ as:

$$T_{i_1 \dots i_n} = \sum_{r_1=1}^{R_1} \dots \sum_{r_n=1}^{R_n} G_{r_1 \dots r_n} U_{i_1 r_1}^{(1)} \dots U_{i_n r_n}^{(n)},$$

where $G \in \mathbb{R}^{R_1 \times \dots \times R_n}$ is the *core tensor* and $U^{(k)} \in \mathbb{R}^{d_k \times R_k}$ are orthogonal *mode matrices*. The string diagram is a star: the core tensor is a central box with n bonds, each connected to a mode matrix.

Categorically, the Tucker decomposition is a morphism in $\mathbf{Para}(\mathbf{FVect}_{\mathbb{R}})$: the parameters are the core tensor and mode matrices (totalling $\prod_k R_k + \sum_k d_k R_k$ numbers), and the implementation is the multilinear contraction. The best rank- $(R_1, \dots, R_n)$ Tucker approximation minimises $\|T - \hat{T}\|_F$ over all such decompositions, and the solution is given by the HOSVD: $U^{(k)}$ consists of the top R_k left singular vectors of the mode- k unfolding of T (the reshaping of T that puts mode k first).

The Tucker decomposition is the categorical foundation of several efficient neural network architectures. A convolutional layer with weight tensor $W \in \mathbb{R}^{k \times k \times C_{\text{in}} \times C_{\text{out}}}$ has Tucker decomposition:

$$W_{ij,c,c'} = \sum_{r,r'} G_{ij,r,r'} U_{c,r}^{(\text{in})} U_{c',r'}^{(\text{out})},$$

which corresponds to three sequential operations: (1) a 1×1 convolution reducing C_{in} channels to R channels ($U^{(\text{in})}$), (2) a $k \times k$ convolution on the reduced channels (G), (3) a 1×1 convolution expanding from R to C_{out} channels ($U^{(\text{out})}$). This is precisely the bottleneck structure of ResNets and MobileNets: the 1×1 convolutions are mode matrices reducing/expanding the channel dimension, and the 3×3 convolution is the core tensor applied to the reduced representation.

32.4 MERA and Hierarchical Representations

The *multi-scale entanglement renormalisation ansatz* (MERA; Vidal, 2007) is a tensor network architecture from quantum many-body physics that represents the wavefunction (or probability distribution) of a system with 2^L sites as a binary tree of tensors. At each level of the tree:

- *Disentangler tensors* $U \in \mathbb{R}^{\chi^2 \times \chi^2}$ (unitary matrices: $U^\top U = I_{\chi^2}$) rearrange the degrees of freedom within each pair of sites, removing short-range correlations.
- *Isometry tensors* $W \in \mathbb{R}^{\chi^2 \times \chi}$ (semi-unitary: $W^\top W = I_\chi$) coarse-grain pairs of sites into single effective sites, moving up the tree.

The MERA is a morphism in the *traced symmetric monoidal category* $(\mathbf{FVect}_{\mathbb{R}}, \otimes, \mathbb{R}, \text{Tr})$ (Chapter 15): the disentanglers are automorphisms $V \otimes V \rightarrow V \otimes V$ and the isometries are morphisms $V \otimes V \rightarrow V$, and the composition along the binary tree uses both the tensor product (horizontal, at each level) and the trace (vertical, when the tree loops back).

The connection to deep learning is the *wavelet scattering network* (Mallat, 2012): replacing the fixed orthogonal transformations of the MERA with learned (but still approximately orthogonal) transformations gives a scattering network that is both multi-scale (processing information at multiple resolutions simultaneously) and equivariant (the orthogonality of the transformations preserves the Lipschitz constant, Chapter 7). The MERA's *renormalisation group* structure (coarse-graining the physical degrees of freedom level by level) corresponds to the wavelet transform's multi-resolution analysis: each level of the MERA corresponds to a wavelet subband, and the MERA approximates the true wavefunction in the same way that a wavelet expansion approximates a function.

32.5 Quantum-Inspired Classical Algorithms

Quantum-inspired algorithms (Tang, 2019) are classical algorithms that achieve runtimes matching the polynomial speedup of certain quantum algorithms, by exploiting the ℓ^2 -norm sampling structure of quantum states. The key insight is that quantum algorithms for linear algebra (recommendation systems, low-rank matrix approximation, linear systems) have classical analogues that achieve the same asymptotic complexity using importance sampling.

The fundamental tool is the ℓ^2 -norm importance sampler: given a vector $v \in \mathbb{R}^n$, sample index i with probability $p_i = v_i^2 / \|v\|^2$ and return the rescaled value $v_i / \sqrt{np_i} = \text{sgn}(v_i) \|v\| / \sqrt{n}$. This gives an unbiased estimator of v_i with variance $\|v\|^4 / (n^2 p_i) - v_i^2 = (n-1)v_i^2 \cdot \|v\|^2 / n^2 + \text{lower order}$.

In categorical terms, the ℓ^2 -norm importance sampler is a morphism in $\mathbf{Para}(\mathbf{BorelStoch})$: it takes a vector $v \in \mathbb{R}^n$ (the parameter) and an index $i \in \{1, \dots, n\}$ (the stochastic output), and returns a random estimate of the i -th coordinate. The morphism maps deterministically to a probability distribution over estimates, making it a Markov kernel in the sense of Chapter 6.

Theorem 32.3 (Tang, 2019; categorical form). *The quantum recommendation system algorithm (Kerenidis-Prakash, 2016) can be dequantised: there exists a classical algorithm in $\mathbf{Para}(\mathbf{BorelStoch})$ that, given ℓ^2 -norm sampling access to the preference matrix $A \in \mathbb{R}^{m \times n}$, computes a low-rank approximation $\hat{A} \approx A_k$ (top- k singular value approximation) in time $O(\text{poly}(k, 1/\epsilon) \cdot \text{poly} \log(mn))$ — matching the quantum speedup. The classical algorithm is a morphism in $\mathbf{Para}(\mathbf{BorelStoch})$ that uses ℓ^2 -norm importance sampling to estimate the singular vectors without accessing the full matrix.*

The categorical significance: the quantum algorithm and the classical quantum-inspired algorithm are two morphisms in **Para(BorelStoch)** that compute the same function (the low-rank approximation) with the same complexity. The quantum algorithm exploits the superposition of quantum states (a compact closed structure specific to quantum mechanics), while the classical algorithm replaces this with ℓ^2 -norm sampling (the classical analogue of quantum amplitude estimation). Both are morphisms in the Markov category, and their equivalence is the statement that the quantum advantage for this specific problem can be achieved classically.

Chapter 33

Categorical Semantics of Programming Languages and Differentiable Programming

33.1 The Curry-Howard-Lambek Correspondence

The deep connection between logic, computation, and category theory is captured by the *Curry-Howard-Lambek correspondence*: a three-way equivalence between intuitionistic propositional logic, the simply typed lambda calculus, and cartesian closed categories. This correspondence is not merely a formal coincidence but a structural identity: the three subjects are the same mathematical object, viewed from different perspectives.

In intuitionistic propositional logic, a *proof* of proposition A from hypotheses Γ is a formal derivation using the rules of the sequent calculus. In the simply typed lambda calculus (STLC), a *term* of type A in context Γ is a well-typed expression using the lambda abstraction and application rules. In a cartesian closed category (CCC), a *morphism* $A \rightarrow B$ is an element of the hom-set $\mathcal{C}(A, B)$. The correspondence is:

Logic	Type Theory	Category Theory
proposition A	type A	object A
proof $\Gamma \vdash A$	term $t : A$ in context Γ	morphism $[\Gamma] \rightarrow A$
implication $A \Rightarrow B$	function type $A \rightarrow B$	exponential $[A \rightarrow B]$
conjunction $A \wedge B$	product type $A \times B$	product $A \times B$
modus ponens	function application	evaluation $\text{ev} : [A \rightarrow B] \times A \rightarrow B$
β -reduction	beta reduction	universal property of ev
η -expansion	eta expansion	uniqueness in universal property

Theorem 33.1 (Lambek-Scott). *The category **STLC** of simply typed lambda calculi (with objects the types and morphisms the $\beta\eta$ -equivalence classes of terms) is equivalent to the category **CCC** of cartesian closed categories (with objects the CCCs and morphisms the CCC-functors). The equivalence sends each STLC to its syntactic CCC (the CCC of types and equivalence classes of terms) and each CCC to the STLC of its internal language.*

The proof of the Lambek-Scott theorem is constructive: given an STLC, the syntactic CCC has types as objects, and the morphisms from Γ to A are $\beta\eta$ -equivalence classes of terms of type A in context Γ . The cartesian structure is given by product types ($\Gamma \times \Delta$ as the context for pairs), the closed structure by function types ($[A \rightarrow B]$ as the exponential), and ev by function application. The coherence conditions (pentagon, triangle) are exactly the $\beta\eta$ -equalities of the STLC.

For machine learning, the Lambek-Scott theorem implies that the denotational semantics of a differentiable programming language (the assignment of mathematical meaning to each program) is a CCC functor, and the correctness of program transformations is the functoriality of this assignment. The correctness of automatic differentiation (Chapter 3) is then the commutativity of a specific CCC functor square, as we develop below.

33.2 Linear Types and the SMCC

The simply typed lambda calculus allows arbitrary copying (via the diagonal $A \rightarrow A \times A$) and discarding (via the terminal map $A \rightarrow \mathbf{1}$) of values. *Linear type theory* (Girard, 1987) restricts this: values must be used exactly once. The corresponding categorical structure is a *symmetric monoidal closed category* (SMCC): a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ in which every pair (A, B) has a *linear exponential* $A \multimap B$ with the evaluation morphism $\text{ev}_\multimap : (A \multimap B) \otimes A \rightarrow B$ and the currying isomorphism $\mathcal{C}(A \otimes B, C) \cong \mathcal{C}(A, B \multimap C)$.

The SMCC is the categorical semantics of *multiplicative intuitionistic linear logic* (MILL): the tensor $\otimes$ models the linear conjunction “both A and B , each used once”, and $A \multimap B$ models the linear implication “consume one A , produce one B ”. The CCC is recovered from the SMCC by adding the *exponential modality* $!$: a comonad $(!, \varepsilon, \delta)$ on the SMCC such that $!A$ can be copied ($\delta_A : !A \rightarrow !A \otimes !A$) and discarded ($\varepsilon_A : !A \rightarrow I$). The Eilenberg-Moore category of the $!$ comonad is a CCC embedded within the SMCC.

For differentiable programming, linear types capture the *resource semantics* of gradient computation. In reverse-mode AD, each intermediate activation is used exactly once in the forward pass (to compute the next layer’s activation) and exactly once in the backward pass (to compute the gradient). Linear types make this “use-once” constraint explicit, and the linear type system statically verifies that no activation is accidentally used twice (which would give incorrect gradients) or discarded without contributing to the gradient (which would give incomplete gradients).

33.3 The Denotational Semantics of AD

The correctness of automatic differentiation as a program transformation can be stated precisely as a commutativity condition in the 2-category of CCCs and CCC-functors.

Define three CCCs:

- $\mathcal{C}_{\text{source}}$: the CCC of the source language (smooth functions $\mathbb{R}^m \rightarrow \mathbb{R}^n$, with the cartesian product monoidal structure and smooth maps as morphisms).

- $\mathcal{C}_{\text{AD}}$: the CCC of the AD language (pairs (f, f') of smooth functions and their derivatives, with the product monoidal structure on pairs).
- $\mathcal{C}_{\text{lens}}$: the lens category $\mathbf{Lens}(\mathcal{C}_{\text{source}})$ (Chapter 3), with lenses $(f, R[f])$ as morphisms.

The denotational semantics $\llbracket \cdot \rrbracket : \mathbf{STLC} \rightarrow \mathcal{C}_{\text{source}}$ assigns to each term its mathematical meaning. The AD transformation $\partial : \mathbf{STLC} \rightarrow \mathbf{STLC}_{\text{AD}}$ is a syntactic transformation that differentiates a source program. The diagram:

$$\begin{array}{ccc} \mathbf{STLC} & \xrightarrow{\llbracket \cdot \rrbracket} & \mathcal{C}_{\text{source}} \\ \partial \downarrow & & \downarrow \text{BP} \\ \mathbf{STLC}_{\text{AD}} & \xrightarrow{\llbracket \cdot \rrbracket_{\text{AD}}} & \mathcal{C}_{\text{lens}} \end{array}$$

commutes if and only if the AD transformation is correct: differentiating a program and then computing its semantics gives the same lens as computing the program's semantics and then applying the backpropagation functor. The commutativity of this diagram is the categorical statement of AD correctness.

Theorem 33.2 (Cruttwell-Gavranović-Ghani-Wilson-Zanasi, 2022). *For any cartesian reverse differential category $\mathcal{C}$ (Chapter 3) and any correct implementation of reverse-mode AD as a functor $\text{BP} : \mathcal{C} \rightarrow \mathbf{Lens}(\mathcal{C})$, the above diagram commutes. Every correct AD library (including PyTorch, JAX, and Zygote) factors through BP via a commutative diagram, and the correctness of any AD library is equivalent to the commutativity of a specific instance of this diagram.*

The proof uses the CRDC axioms (Chapter 3): the chain rule axiom ($R[g \circ f](x, c') = R[f](x, R[g](f(x), c'))$) is exactly the functoriality of BP, and the remaining CRDC axioms (linearity, additivity, projection) ensure that BP is a symmetric monoidal functor. The commutativity of the diagram follows from the CCC structure of both the source language and the lens category.

33.4 Effects, Monads, and Side Effects in Differentiable Programs

Real differentiable programming languages involve computational effects: random number generation (for stochastic layers), I/O (for data loading), and mutation (for in-place parameter updates). The categorical semantics of effects uses monads (Chapter 13) to encapsulate them.

The *state monad* $T_S(A) = S \rightarrow (A \times S)$ (for a state space S , e.g., the space of random seeds or parameter arrays) encapsulates stateful computation: a morphism $A \rightarrow T_S(B)$ is a function that takes an input A and a current state S , and produces an output B and a new state S' . The Kleisli category of T_S is the category of stateful programs, and Kleisli composition sequences state transformations. For differentiable programs, differentiating through stateful computations requires computing the gradient with respect to both the input and the state update, which is handled by the lens composition of the state monad

with the CRDC structure.

The *probability monad* $\mathcal{P}$ (Chapter 13) encapsulates randomness. A morphism $A \rightarrow \mathcal{P}(B)$ is a stochastic program that, given an input A , produces a random output in B . Differentiating a stochastic program requires computing the gradient of the expected output. Two main strategies are:

Reparametrisation trick: if the random variable $z \sim p_\theta(z)$ can be written as $z = g_\theta(\varepsilon)$ with $\varepsilon \sim p(\varepsilon)$ a fixed reference distribution, then:

$$\nabla_\theta \mathbb{E}_{z \sim p_\theta} [f(z)] = \mathbb{E}_{\varepsilon \sim p} [\nabla_\theta f(g_\theta(\varepsilon))] = \mathbb{E}_\varepsilon [\nabla_z f \cdot \nabla_\theta g_\theta(\varepsilon)].$$

Categorically, this is the pullback of the gradient along the reparametrisation morphism $g_\theta : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(Z)$: the gradient in the reparametrised space is pulled back to the original space via the chain rule. The reparametrisation trick is thus a morphism in the CRDC of the probability monad’s Kleisli category.

REINFORCE estimator: for discrete or non-reparametrisable distributions:

$$\nabla_\theta \mathbb{E}_{z \sim p_\theta} [f(z)] = \mathbb{E}_{z \sim p_\theta} [f(z) \nabla_\theta \log p_\theta(z)].$$

This is the score natural transformation $s : \mathcal{G}(M) \Rightarrow \mathfrak{X}(M)$ (Chapter 23) applied to p_θ and composed with the expectation of $f(z)$: it uses the score function $\nabla_\theta \log p_\theta(z)$ as the “gradient direction” and $f(z)$ as the “weight” for each sample. The REINFORCE estimator is a morphism in **Para(BorelStoch)**.

33.5 Abstract Interpretation and the Categorical Semantics of Verification

Abstract interpretation (Cousot-Cousot, 1977) is a framework for static analysis of programs: instead of executing the program on all possible inputs (infeasible), we execute an *abstract version* on an abstract domain that over-approximates the set of possible values. For neural network verification, the abstract domain represents sets of possible activations (not single activations), and the abstract semantics over-approximates the true set of outputs.

The categorical framework is the following. Let $\mathcal{C}$ be the CCC of the source language (e.g., **Smooth** for neural networks). An *abstract domain* is a CCC $\mathcal{A}$ together with a *Galois connection* $(\alpha : \mathcal{C} \rightarrow \mathcal{A}, \gamma : \mathcal{A} \rightarrow \mathcal{C})$ satisfying:

- α is a left adjoint to γ : $\mathcal{A}(\alpha(c), a) \cong \mathcal{C}(c, \gamma(a))$ for all $c \in \mathcal{C}$, $a \in \mathcal{A}$.
- α preserves the relevant structure (products, exponentials).

The abstract semantics $\llbracket f \rrbracket^\# : \mathcal{A} \rightarrow \mathcal{A}$ of a program $f : \mathcal{C} \rightarrow \mathcal{C}$ is the morphism $\llbracket f \rrbracket^\# = \alpha \circ \llbracket f \rrbracket \circ \gamma$, computing the tightest over-approximation of f ’s output consistent with the abstract domain.

Soundness of abstract interpretation is the condition:

$$\alpha(\llbracket f \rrbracket(c)) \leq \llbracket f \rrbracket^\#(\alpha(c)) \quad \text{for all } c \in \mathcal{C},$$

where $\leq$ is the order relation in $\mathcal{A}$ (the abstract domain is a partially ordered set, and “ $a \leq b$ ” means “ a is a more precise description than b ”). This is equivalent to the condition that the following diagram commutes *laxly* (i.e., the square commutes up to a 2-cell in the order):

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{\llbracket f \rrbracket} & \mathcal{C} \\ \alpha \downarrow & \lhd \leq_{\mathcal{A}} & \downarrow \alpha \\ \mathcal{A} & \xrightarrow{\llbracket f \rrbracket^\#} & \mathcal{A} \end{array}$$

The Galois connection (α, γ) ensures soundness: by the adjunction, $\alpha \circ \llbracket f \rrbracket = \alpha \circ \llbracket f \rrbracket \circ \gamma \circ \alpha = \llbracket f \rrbracket^\# \circ \alpha$ (using the unit of the adjunction $\text{id} \leq \gamma \circ \alpha$). The abstract semantics is thus the best (most precise) sound over-approximation of the true semantics consistent with the abstract domain.

Concrete abstract domains for neural network verification include:

- *Interval arithmetic*: $\alpha(f) = [\min_x f(x), \max_x f(x)]$ over bounded input sets. The abstract semantics of a ReLU layer applied to an interval $[l, u]$ is $[\max(0, l), \max(0, u)]$ (interval propagation). Efficient: polynomial time in the number of neurons.
- *Zonotopes*: $\alpha(f) = \{c + \sum_i \varepsilon_i e_i : \varepsilon_i \in [-1, 1]\}$ (ellipsoidal approximations). More precise than intervals: can represent linear correlations between coordinates. Cost: $O(n^2)$ per layer.
- *Semidefinite programming relaxations*: $\alpha(f)$ = the convex hull of the image of f over the input set. Most precise: can verify tighter bounds. Cost: $O(n^3)$ per layer (SDP solver).

Each corresponds to a different Galois connection between $\mathcal{C}$ and a specific abstract domain $\mathcal{A}$, and the tradeoff between precision and efficiency is the tradeoff between the tightness of the Galois connection and the computational cost of the abstract semantics.

Concluding Remarks

This monograph has developed, across thirty-three chapters, the thesis that category theory is not merely a language for describing deep learning but a theory that explains it: a framework in which the fundamental operations of machine learning — composition of layers, backpropagation, probabilistic inference, symmetry-equivariant computation, optimal transport, causal reasoning, and the dynamics of gradient-based optimisation — arise as instances of universal categorical constructions, and in which the correctness, compositionality, and robustness of these operations are consequences of the categorical axioms.

The journey began with the bare definitions of Chapter 1 (categories, functors, natural transformations) and proceeded through the monoidal structure necessary to describe parallel computation (Chapter 2), the lens and optic framework that gives backpropagation its canonical categorical form (Chapters 3–4), polynomial functors and their role in recurrent architectures (Chapter 5), the Markov category framework for probability (Chapter 6), enriched categories and Lipschitz robustness (Chapter 7), the Yoneda lemma and the geometry of representation (Chapter 8), and the topos-theoretic account of compositional generalisation (Chapter 9). The middle portion of the monograph developed the applied categorical theory: equivariant architectures as G -equivariant morphisms (Chapter 18), the information geometry of the natural gradient (Chapter 19), the signal-processing interpretation of convolution as a monoidal functor (Chapter 20), the free energy principle as a fixed point in the lens category (Chapter 21), and the categorical classification of loss functions via proper scoring rules and Bregman divergences (Chapter 22). The final portion — Chapters 23 through 33 — treated the major architecture families (generative models, residual networks, graph neural networks, Transformers, reinforcement learning, scaling laws, tensor networks, and differentiable programming) each from a categorical perspective that reveals structure invisible to the standard mathematical treatments.

Several themes have recurred throughout and deserve emphasis as synthesising principles.

Functoriality as correctness. The most important recurring theme is that the correctness of compositional algorithms is equivalent to the functoriality of a specific map between categories. The chain rule of calculus is the functoriality of the tangent functor $T : \mathbf{Smooth} \rightarrow \mathbf{Smooth}$ (Chapter 1). The correctness of backpropagation is the functoriality of the lens composition (Chapter 3) and the backpropagation functor $\mathbf{BP} : \mathbf{Para}(\mathbf{Smooth}) \rightarrow \mathbf{Lens}(\mathbf{Smooth})$ (Chapters 3–4). The correctness of MAML’s second-order gradient is the functoriality of the 2-functor $\mathbf{BP}_2$ (Chapter 26). The correctness of score matching is the naturality of the score transformation $s : \mathcal{G}(M) \Rightarrow \mathfrak{X}(M)$ (Chap-

ter 23). The correctness of automatic differentiation libraries (JAX, PyTorch, Zygote) is the commutativity of a specific CCC functor diagram (Chapter 33). In each case, what might appear to be an ad hoc computational fact is revealed as a consequence of a universal categorical property.

Universality as optimality. The second theme is that universal categorical constructions — limits, colimits, Kan extensions, adjunctions — characterise optimal solutions to learning problems. The Yoneda lemma identifies the optimal representation of a class (the representable functor) and justifies the prototypical network classifier (Chapter 8). Kan extensions formalise generalisation from training data to unseen inputs, with the choice of enrichment determining the inductive bias (Chapter 12). The Wasserstein distance is the enriched hom-object in the Lawvere metric category of probability distributions (Chapter 7), making it the canonical distance for optimal transport (Chapter 23). The Chinchilla scaling optimum is the terminal object in the category of compute-constrained allocations (Chapter 31). In each case, the categorical characterisation not only describes what the optimal solution is, but explains *why* it is optimal: it is the unique object with the universal property.

Duality as bidirectionality. The third theme is that the dual of a categorical construction captures the backward direction of computation. The opposite category $\mathcal{C}^{\text{op}}$ reverses morphisms, capturing the reversal from the forward to the backward pass. Lenses and optics are the categorical structures that package forward and backward together, and their composition law is the chain rule (Chapter 3). The Tambara representation theorem (Chapter 4) shows that optics are the universal such structure: any reasonable bidirectional process factors through the optic. The adjoint method for Neural ODEs (Chapter 24) is the categorical dual of the forward ODE solve. The actor-critic architecture in RL (Chapter 30) is a bidirectional morphism in **BorelStoch**: the actor computes forward (from state to action) and the critic computes backward (from reward to value estimate). The Curry-Howard correspondence (Chapter 33) pairs propositions with their proofs in exactly the same way that types are paired with their terms, which is exactly how objects are paired with their morphisms.

Coherence as compiler guarantee. The fourth theme is that the coherence theorems of category theory (Mac Lane’s coherence for monoidal categories, Joyal-Street for string diagrams, Tambara representation for optics, completeness of the do-calculus in Markov categories) are compiler guarantees: they ensure that every mathematically valid composition of operations produces a consistent and well-defined result, regardless of the order in which the operations are applied. A deep learning framework built on coherent categorical foundations would, by construction, be free of the compositional inconsistencies that require ad hoc verification in current systems.

The open problems. Several significant open problems remain.

The *homotopy-coherent Para construction* would make the backpropagation functor fully $(\infty, 1)$ -natural, ensuring that all reparametrisation equivalences are coherently tracked at every level of the categorical hierarchy (Chapter 16).

The *categorical semantics of in-context learning* would explain how a Transformer trained by gradient descent on a fixed distribution can solve new tasks at inference time by reading task demonstrations from its context window, without updating its weights. This phenomenon has no satisfactory categorical account and is arguably the most important structural mystery in contemporary deep learning.

The *categorical theory of normalisation layers* would clarify how BatchNorm, LayerNorm, and their variants interact with the Para construction and the backpropagation functor, providing a rigorous account of their role in training stability and their effect on the geometry of the loss landscape.

The *formal verification of Transformer correctness* would provide a machine-checked proof, in Agda or Lean, that the attention mechanism is equivariant, that the positional encoding satisfies the appropriate coherence conditions with respect to sequence ordering, and that the backpropagation functor computes the correct gradient for the full Transformer block.

The *categorical account of emergent capabilities* would explain why certain capabilities (multi-step reasoning, arithmetic, code generation) appear suddenly at critical model scales as discrete phase transitions rather than smooth improvements, connecting the Morse theory of the loss landscape (Chapter 27) to the information geometry of the scaling laws (Chapter 31).

The central lesson. Category theory entered mathematics as a language for algebraic topology, matured into a foundation for logic and computer science, and has now arrived at machine learning. Its arrival is not accidental. Deep learning — like every mature mathematical subject — has an intrinsic structure that becomes visible only when viewed at the right level of abstraction. That level is the level of categories, functors, and natural transformations. The central lesson of this monograph, stated in one sentence, is the oldest lesson in mathematics: the right language for a subject is not a convenience but a necessity, because it determines what can be proved, what can be computed, and what questions it is even possible to ask.

Exercises

The following exercises are organised by chapter. They range from direct verifications of definitions and examples to more substantial problems that require combining ideas from multiple chapters. Exercises marked with $(\star)$ are more challenging and may require consulting the references in the Bibliographic Notes.

Chapter 1. Categories, Functors, and Natural Transformations.

1. Verify that the category **Grp** of groups and group homomorphisms satisfies the category axioms. What is the identity morphism at a group G ? What is the composition of two group homomorphisms?
2. Show that a poset $(P, \leq)$ defines a category: identify the objects and morphisms, and verify associativity and the unit laws.
3. Prove that the tangent functor $T : \mathbf{Smooth} \rightarrow \mathbf{Smooth}$ satisfies the functoriality conditions $T(\text{id}_M) = \text{id}_{TM}$ and $T(g \circ f) = T(g) \circ T(f)$ for all smooth maps $f : M \rightarrow N$ and $g : N \rightarrow K$. What familiar theorem in differential calculus does the second condition express?
4. Let $F : \mathcal{C} \rightarrow \mathcal{D}$ and $G : \mathcal{D} \rightarrow \mathcal{E}$ be functors. Define the *composite functor* $G \circ F : \mathcal{C} \rightarrow \mathcal{E}$ and verify that it is indeed a functor. Show that the composite of two natural transformations $\eta : F \Rightarrow F'$ and $\mu : G \Rightarrow G'$ gives a natural transformation $\mu * \eta : G \circ F \Rightarrow G' \circ F'$ (the *horizontal composition* of natural transformations).
5. Show that the category $\mathbf{Mat}_{\mathbb{R}}$ (objects are natural numbers $n \in \mathbb{N}$; morphisms $m \rightarrow n$ are $n \times m$ real matrices; composition is matrix multiplication) is isomorphic to the full subcategory of $\mathbf{FVect}_{\mathbb{R}}$ whose objects are the standard spaces $\{\mathbb{R}^n : n \in \mathbb{N}\}$.
6. $(\star)$ Show that an adjunction $F \dashv G$ between categories $\mathcal{C}$ and $\mathcal{D}$ determines a monad $(G \circ F, \eta, G\varepsilon_F)$ on $\mathcal{C}$, where $\eta : \text{Id}_{\mathcal{C}} \Rightarrow G \circ F$ is the unit and $\varepsilon : F \circ G \Rightarrow \text{Id}_{\mathcal{D}}$ is the counit of the adjunction. Verify the monad axioms from the triangle identities of the adjunction.

Chapter 2. Monoidal Categories and String Diagrams.

1. Draw the string diagram for the composition $h \circ (g \otimes f) : A \otimes C \rightarrow D$ where $f : C \rightarrow B$, $g : A \rightarrow A'$, and $h : A' \otimes B \rightarrow D$ are morphisms in a symmetric monoidal category. Label all wires and boxes.

2. Verify that $(\mathbf{Set}, \times, \{*\})$ (sets with cartesian product and a singleton as unit) is a symmetric monoidal category. What are the natural isomorphisms α , λ , ρ ? Write out the pentagon equation for four sets A, B, C, D and verify it holds.
3. Show that $\mathbf{FVect}_{\mathbb{R}}$ with the tensor product is compact closed by: (a) defining $\eta_V : \mathbb{R} \rightarrow V^* \otimes V$ for a basis $\{e_i\}$ of V ; (b) defining $\varepsilon_V : V \otimes V^* \rightarrow \mathbb{R}$; (c) verifying the snake equations. What goes wrong if you try to make $\mathbf{Set}$ compact closed?
4. Express the matrix transpose $W^\top$ of a linear map $W : V \rightarrow W$ as a string diagram in $\mathbf{FVect}_{\mathbb{R}}$ using the unit and counit of the compact closed structure. Verify that the string diagram correctly computes $(W^\top)_{ij} = W_{ji}$.
5. $(\star)$ The *Frobenius law* for a bimonoid $(A, \nabla, \eta, \Delta, \varepsilon)$ states $(\nabla \otimes \text{id}_A) \circ (\text{id}_A \otimes \Delta) = \Delta \circ \nabla = (\text{id}_A \otimes \nabla) \circ (\Delta \otimes \text{id}_A)$. Draw this as a string diagram and verify it holds for $A = \mathbb{R}^d$ with $\nabla(u, v) = u + v$ and $\Delta(v) = (v, v)$.

Chapter 3. Lenses, Para, and Backpropagation as a Functor.

1. Write out the lens composition formula explicitly for two lenses $\ell_1 = (\mathbf{get}_1, \mathbf{put}_1) : (S_1, A_1) \rightarrow (S_2, A_2)$ and $\ell_2 = (\mathbf{get}_2, \mathbf{put}_2) : (S_2, A_2) \rightarrow (S_3, A_3)$. Verify that the composition satisfies the left unit law: $\text{id}_{(S,A)} \circ \ell = \ell$.
2. For a differentiable function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, compute the lens $\text{BP}(f) = (f, R[f])$ explicitly. What is $\mathbf{get}(\theta, x)$? What is $\mathbf{put}((\theta, x), \delta y)$ for $\delta y \in \mathbb{R}^n$? Check the chain rule: $\text{BP}(g \circ f) = \text{BP}(g) \circ \text{BP}(f)$ for $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^k$.
3. Consider the Para construction $\mathbf{Para}(\mathbf{FVect}_{\mathbb{R}})$. A morphism $(P, f) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is a linear map $f : P \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ (bilinear in p and x). Show that every such morphism can be written as $f(p, x) = Ax$ where $A \in \mathbb{R}^{n \times m}$ depends linearly on $p \in P$. What is the composition of two such morphisms in $\mathbf{Para}(\mathbf{FVect}_{\mathbb{R}})$?
4. Verify that the CRDC chain rule axiom $R[g \circ f](x, c') = R[f](x, R[g](f(x), c'))$ reduces to the classical chain rule $(J_{g \circ f}(x))^\top c' = (J_f(x))^\top (J_g(f(x)))^\top c'$ when $R[f](x, \delta y) = (J_f(x))^\top \delta y$.
5. $(\star)$ Show that the lens category $\mathbf{Lens}(\mathcal{C})$ has a symmetric monoidal structure: define the tensor product of two lenses $\ell_1 : (S_1, A_1) \rightarrow (T_1, B_1)$ and $\ell_2 : (S_2, A_2) \rightarrow (T_2, B_2)$, and verify the associativity and unit laws. What does this monoidal structure correspond to in the backpropagation interpretation?

Chapter 4. Optics, Tambara Modules, and Automatic Differentiation.

1. Verify that every lens (in a cartesian category) is an optic: write $(\mathbf{get}, \mathbf{put}) : (S, T) \rightarrow (A, B)$ as an element of the coend $\int^M \mathcal{C}(S, M \times A) \times \mathcal{C}(M \times B, T)$ by choosing $M = S$, $l = \Delta_S = s \mapsto (s, s)$ (the diagonal), and $r = \mathbf{put}$.
2. Show that the cowedge condition for the optic coend identifies two representations (M, l, r) and (M', l', r') when there is a morphism $h : M \rightarrow M'$ such that $l' = (h \times \text{id}_A) \circ l$ and $r = r' \circ (h \times \text{id}_B)$. Why does this identification make the optic representation basis-independent?

3. For the exponential kernel $K(q, k) = \exp(q \cdot k / \sqrt{d})$ used in attention, compute the feature map ϕ such that $K(q, k) = \langle \phi(q), \phi(k) \rangle_{\mathcal{H}_K}$ in the RKHS $\mathcal{H}_K$ (hint: use the Taylor expansion of the exponential and the multinomial theorem). Why is this feature map infinite-dimensional?
4. (★) State and prove the Ninja Yoneda Lemma: $\int^{C \in \mathcal{C}} \mathcal{C}(C, A) \times G(C) \cong G(A)$ naturally in A . Hint: use the regular Yoneda lemma applied to the coend expression and the fact that $\mathcal{C}(C, A)$ is representable.

Chapter 5. Polynomial Functors and Neural Dynamical Systems.

1. Write out the polynomial functor $p(Y) = \mathcal{G}(S) \times (I \rightarrow Y)$ corresponding to a Markov decision process with state space S and action space I . What are the positions of p ? What are the directions at each position?
2. Show that the category of polynomial functors **Poly** has a monoidal structure given by composition: $(p \otimes q)(Y) = p(q(Y))$. Verify that composition of polynomials corresponds to the sequential coupling of two dynamical systems (the output of the inner system drives the input of the outer system).
3. For the RNN cell $f : H \times X \rightarrow Y \times H$ (where H is the hidden state space, X the input space, and Y the output space), write the corresponding polynomial morphism $\phi : H \cdot y^H \rightarrow Y \cdot y^X$ in **Poly**. What are ϕ_1 and $\phi^\#$?
4. (★) Show that the final coalgebra of the polynomial functor $O \cdot y^I$ (on **Set**) is the set of all functions $I^\omega \rightarrow O^\omega$ from infinite input streams to infinite output streams. Hint: use the fact that the final coalgebra of a functor F is the set of all F -behaviours, and that a behaviour for $O \cdot y^I$ is an infinite tree.

Chapter 6. Categorical Probability and Markov Categories.

1. Verify that the category **BorelStoch** of standard Borel spaces and Markov kernels satisfies the Markov category axioms: (a) composition is well-defined (the Chapman-Kolmogorov equation); (b) the copy morphism $\delta_X : X \rightarrow X \times X$, $\delta_X(A \times B \mid x) = \mathbf{1}_{x \in A} \mathbf{1}_{x \in B}$ is a natural transformation; (c) the deletion morphism $\varepsilon_X : X \rightarrow \{*\}$ satisfies $\varepsilon_X \circ f = \varepsilon_Y$ for all Markov kernels $f : X \rightarrow Y$.
2. Express the conditional independence $X \perp\!\!\!\perp Y \mid Z$ in the string diagram calculus of **BorelStoch**: draw the factorisation condition $f = (f_{X|Z} \otimes f_{Y|Z}) \circ \delta_Z \circ f_Z$ where $f : * \rightarrow X \times Y \times Z$ is the joint distribution.
3. Show that the Giry monad $\mathcal{G}$ on **Meas** has the monad axioms: (a) the unit $\eta_X(A \mid x) = \mathbf{1}_{x \in A}$ (Dirac delta) satisfies $\mu \circ \eta_{GX} = \text{id}_{GX}$ and $\mu \circ G\eta_X = \text{id}_{GX}$; (b) the multiplication $\mu_X : \mathcal{G}(\mathcal{G}(X)) \rightarrow \mathcal{G}(X)$ (integration of a measure over a measure) satisfies the associativity condition $\mu \circ G\mu = \mu \circ \mu_G$.
4. (★) Show that the Kleisli category of the Giry monad is the category **Stoch** of Markov kernels. Specifically, show that: (a) objects of **Meas_G** are measurable spaces; (b) morphisms $X \rightarrow Y$ in **Meas_G** are Markov kernels $k : X \rightarrow \mathcal{G}(Y)$; (c) Kleisli composition $k_2 \circ_{\mathcal{G}} k_1 = \mu_Z \circ \mathcal{G}(k_2) \circ k_1$ is the Chapman-Kolmogorov equation.

Chapter 7. Enriched Categories, Metric Spaces, and Robustness.

1. Verify Lawvere's theorem: a generalised metric space (X, d) (with $d : X \times X \rightarrow [0, \infty]$, $d(x, x) = 0$, and $d(x, z) \leq d(x, y) + d(y, z)$) defines a $[0, \infty]$ -enriched category with $\text{ob}(\mathcal{C}) = X$ and $\mathcal{C}(x, y) = d(x, y)$. Check all enriched category axioms.
2. A neural network layer $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ with Lipschitz constant L is a morphism in the enriched category of metric spaces from $(\mathbb{R}^m, L^{-1}\|\cdot\|)$ to $(\mathbb{R}^n, \|\cdot\|)$. Show that the composition of two such layers with Lipschitz constants L_1 and L_2 has Lipschitz constant at most $L_1 L_2$. This is the enriched functor composition law.
3. For a matrix $W \in \mathbb{R}^{n \times m}$, show that its spectral norm $\|W\|_2 = \sigma_{\max}(W)$ (the largest singular value) is the Lipschitz constant of the linear map $x \mapsto Wx$. Hence spectral normalisation $W \leftarrow W/\|W\|_2$ ensures that each linear layer is 1-Lipschitz.
4. $(\star)$ Show that the p -Wasserstein distance $W_p(\mu, \nu)$ on probability measures over a metric space (X, d) is the hom-object of a Lawvere metric on $\mathcal{G}(X)$: verify (a) $W_p(\mu, \mu) = 0$; (b) $W_p(\mu, \nu) \geq 0$; (c) $W_p(\mu, \nu) + W_p(\nu, \rho) \geq W_p(\mu, \rho)$ (triangle inequality). The triangle inequality follows from the coupling argument: the composition of two optimal couplings is a valid (but not necessarily optimal) coupling.

Chapter 8. The Yoneda Lemma and the Geometry of Representation.

1. Prove the Yoneda Lemma directly: for a functor $F : \mathcal{C} \rightarrow \mathbf{Set}$ and object $A \in \mathcal{C}$, construct the bijection $\text{Nat}(\mathcal{C}(A, -), F) \cong F(A)$ and show it is natural in A and F by checking the relevant naturality squares.
2. Show that the Yoneda embedding $\mathcal{Y} : \mathcal{C}^{\text{op}} \rightarrow [\mathcal{C}, \mathbf{Set}]$, $A \mapsto \mathcal{C}(A, -)$, is *fully faithful*: the induced map $\mathcal{C}(A, B) \rightarrow \text{Nat}(\mathcal{C}(B, -), \mathcal{C}(A, -))$ is a bijection. Conclude that two objects are isomorphic in $\mathcal{C}$ if and only if their Yoneda embeddings are naturally isomorphic.
3. The skip-gram objective of word2vec minimises $-\sum_{(w,c)} \log \sigma(v_w \cdot v_c) - \sum_{(w,c')} \log \sigma(-v_w \cdot v_{c'})$, where (w, c) are co-occurring word-context pairs and (w, c') are negative samples. Show that the solution to this objective can be written as a low-rank factorisation of the pointwise mutual information matrix $\text{PMI}(w, c) = \log(p(w, c)/(p(w)p(c)))$. Interpret this factorisation in terms of the Yoneda embedding of words into the co-occurrence category.
4. $(\star)$ Show that for a Gaussian kernel $K(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$, the neural tangent kernel $K_\theta(x, x') = \langle \nabla_\theta f_\theta(x), \nabla_\theta f_\theta(x') \rangle$ of an infinitely wide neural network converges (at initialisation) to a fixed kernel $K_\infty(x, x')$ that depends on the architecture and activation function but not on the specific random initialisation. Interpret K_∞ as an inner product of Yoneda representations of the data points in the RKHS defined by the tangent kernel.

Bibliographic Notes and References

The following notes are organised by chapter. They identify the primary sources for the results presented, situate each topic within its broader literature, and provide pointers for further reading. Where multiple papers address the same result, we cite the most accessible or influential version.

General references. The standard reference for category theory is Mac Lane’s *Categories for the Working Mathematician* (Springer, 2nd ed., 1998). Riehl’s *Category Theory in Context* (Dover, 2016) is more pedagogical and free online. Borceux’s three-volume *Handbook of Categorical Algebra* (Cambridge, 1994) is encyclopedic. For enriched category theory, Kelly’s *Basic Concepts of Enriched Category Theory* (Cambridge, 1982; also freely available at TAC) remains definitive. Leinster’s *Basic Category Theory* (Cambridge, 2014; arXiv:1612.09375) is an excellent short introduction.

For the applications to machine learning, Fong and Spivak’s *An Invitation to Applied Category Theory* (Cambridge, 2019; arXiv:1803.05316) is the essential companion, covering many of the monoidal and compositional structures used throughout this monograph. Gavranović’s doctoral thesis *Categorical Deep Learning: An Algebraic Theory of Architectures* (University of Zagreb, 2024; arXiv:2402.15332) is the most complete single treatment of the categorical theory of backpropagation and parametric learning and is the primary reference for Chapters 3, 4, 14, 15, and 26.

Chapter 1. Categories, Functors, and Natural Transformations. The definitions are standard; see Mac Lane (1998) or Riehl (2016). The free vector space functor and the one-hot encoding connection are discussed in Fong-Spivak (2019). The tangent functor as the categorical expression of the chain rule appears in Cockett-Cruttwell, *Differential structure, tangent structure, and SDG*, Applied Categorical Structures 22 (2014), 331–417, and in Cruttwell’s doctoral thesis (Dalhousie, 2008).

Chapter 2. Monoidal Categories and String Diagrams. The coherence theorem for monoidal categories is Mac Lane’s 1963 result; see Mac Lane (1998), Chapter XI. The Joyal-Street theorem on string diagrams (topological correctness) appears in Joyal-Street, *The Geometry of Tensor Calculus I*, Advances in Mathematics 88 (1991), 55–112. Coecke-Kissinger’s *Picturing Quantum Processes* (Cambridge, 2017) gives the most thorough treatment of string diagrams in compact closed categories. The **Learn** category of Fong-Spivak-Tuyéras appears in *Backprop as a Functor* (LICS 2019, arXiv:1711.10455).

Chapter 3. Lenses, Para, and Backpropagation as a Functor. The lens framework originates in functional programming; see van Laarhoven (2009) and O'Connor (2011). The categorical treatment is due to Johnson-Rosebrugh-Wood, *Lenses, fibrations and universal translations*, Mathematical Structures in Computer Science 22 (2012), 25–42. The Para construction and the backpropagation functor are developed in Cruttwell-Gavranović-Ghani-Wilson-Zanasi, *Categorical Foundations of Gradient-Based Learning*, ESOP 2022 (arXiv:2103.01931). The cartesian reverse differential category (CRDC) axioms are from Cockett-Cruttwell, *Reverse Derivative Categories*, CSL 2019 (arXiv:1910.07065).

Chapter 4. Optics, Tambara Modules, and Automatic Differentiation. The optic as a coend is Riley's *Categories of Optics* (2018, arXiv:1809.00738). The Tambara module representation theorem is in Boisseau-Gibbons, *What You Needa Know about Yoneda* (ICFP 2018) and Clarke-Elkins-Gibbons-Zardini-Milewski-Zanasi, *Profunctor Optics: A Categorical Update*, Compositionality 6 (2024). The application to reverse-mode AD is developed in Cruttwell et al. (ESOP 2022) and in Nunes, *Categorical Foundations of Gradient-Based Learning* (Master's thesis, IST Lisbon, 2021). Elliott's *The simple essence of automatic differentiation* (ICFP 2018, arXiv:1804.00746) gives the functional programming perspective.

Chapter 5. Polynomial Functors and Neural Dynamical Systems. Spivak's *Polynomial Functors: A Mathematical Theory of Interaction* (2022, arXiv:2312.00990) and the book with Niu, *Polynomial Functors* (2023) are the foundational references. The interpretation of dynamical systems as polynomial coalgebras is developed in Spivak-Myers, *Double Categories of Open Dynamical Systems* (MFPS 2020, arXiv:2005.05956) and Libkind-Baas-Halter-Patterson-Fairbanks, *Operadic Modeling of Dynamical Systems* (2022, arXiv:2105.12282). The connection to SSMs (including Mamba) is explored in several 2023–2024 preprints; see also Gu-Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces* (2023, arXiv:2312.00752).

Chapter 6. Categorical Probability and Markov Categories. The Markov category framework is Fritz's *A Synthetic Approach to Markov Kernels, Conditional Independence and Theorems on Sufficient Statistics*, Advances in Mathematics 370 (2020), 107239 (arXiv:1908.07021). This paper is the essential reference for Chapters 6, 14, 19, 25, and 29. The Giry monad originates in Giry, *A categorical approach to probability theory*, Lecture Notes in Mathematics 915 (1982), 68–85. Fritz-Liang, *Free gs-monoidal categories and free Markov categories*, Applied Categorical Structures 31 (2023, arXiv:2204.02284) develops the free constructions.

Chapter 7. Enriched Categories, Metric Spaces, and Robustness. Lawvere's identification of metric spaces as enriched categories is in *Metric Spaces, Generalised Logic, and Closed Categories*, Rendiconti del Seminario Matematico e Fisico di Milano 43 (1973), 135–166 (reprinted in TAC). Quantale-enriched categories and their applications to logic and metric geometry are surveyed in Hofmann-Seal-Tholen (eds.), *Monoidal Topology* (Cambridge, 2014). The application to Lipschitz robustness of neural networks is pursued in

Tsuzuku-Sato-Sugiyama, *Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks*, NeurIPS 2018.

Chapter 8. The Yoneda Lemma and the Geometry of Representation. The Yoneda lemma is in Mac Lane (1998), Chapter III. The word2vec model is Mikolov et al., *Distributed Representations of Words and Phrases and their Compositionality*, NeurIPS 2013. The neural tangent kernel (NTK) is Jacot-Gabriel-Hongler, *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*, NeurIPS 2018 (arXiv:1806.07572). The Yoneda-NTK connection is implicit in the NTK paper and made explicit in several follow-up works; see Lee et al., *Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent*, NeurIPS 2019.

Chapter 9. Topoi, Sheaves, and the Logic of Generalisation. The theory of Grothendieck topoi is developed in Mac Lane-Moerdijk, *Sheaves in Geometry and Logic* (Springer, 1992) and Johnstone’s *Sketches of an Elephant* (Oxford, 2002). The application of sheaf theory to compositional generalisation in neural networks is proposed in Grefenstette et al. and developed formally in Toumi-Pivkin-Coecke, *Diagrammatic Differentiation for Quantum Machine Learning*, QPL 2021 (arXiv:2103.07960). The connection to the Mitchell-Bénabou language is covered in Lambek-Scott, *Introduction to Higher Order Categorical Logic* (Cambridge, 1986).

Chapter 10. Categorical Cybernetics and Open Games. The original open game paper is Ghani-Hedges-Winschel-Zahn, *Compositional Game Theory*, LICS 2018 (arXiv:1603.04641). Hedges’s doctoral thesis (Queen Mary, 2016) gives a comprehensive treatment. The categorical cybernetics framework unifying lenses, games, and learning is in Capucci-Gavranović-Hedges-Rischel, *Towards Foundations of Categorical Cybernetics*, MFPS 2022 (arXiv:2105.06332). The GAN game (Chapter 23) is Goodfellow et al., *Generative Adversarial Networks*, NeurIPS 2014 (arXiv:1406.2661). SAM (Chapter 27) is Foret-Kleiner-Mobahi-Neyshabur, *Sharpness-Aware Minimization for Efficiently Improving Generalization*, ICLR 2021 (arXiv:2010.01412).

Chapter 11. Free Probability, the Composition of Random Layers, and the Freeness Conjecture. The foundational reference for free probability is Voiculescu-Dykema-Nica, *Free Random Variables* (AMS, 1992). Mingo-Speicher’s *Free Probability and Random Matrices* (Springer, 2017) is a comprehensive modern treatment. The application to deep networks is initiated in Pennington-Wakhloo, *Geometry of Neural Network Loss Surfaces via Random Matrix Theory*, ICML 2017, and developed in Benigni-Péché, *Eigenvalue Distribution of Nonlinear Models of Random Matrices* (2022, arXiv:1904.03090). The freeness conjecture is stated clearly in Pennington-Schoenholz-Ganguli, *The Emergence of Spectral Universality in Deep Networks*, AISTATS 2018.

Chapter 12. Categorical Limits, Colimits, and Universal Approximation. Limits and colimits are covered in Mac Lane (1998), Chapter V. The universal approximation theorem for neural networks with sigmoid activations is Cybenko (1989) and Hornik-Stinchcombe-White (1989); for ReLU networks see Leshno et al. (1993). The Kan extension

formulation of generalisation is discussed in Fong-Spivak-Tuyéras (LICS 2019). The connection to RKHS and minimum-norm interpolation is developed in the kernel learning literature; see Schölkopf-Smola, *Learning with Kernels* (MIT Press, 2002).

Chapter 13. Monads, Comonads, and the Algebra of Effects. Monads in category theory are from Eilenberg-Moore (1965) and Kleisli (1965). The identification of monads with “monoids in the category of endofunctors” is Mac Lane (1998), Chapter VI. The Gir monad and probabilistic programming are in Staton, *Commutative Semantics for Probabilistic Programming*, ESOP 2017. The stream comonad and convolutional networks are discussed in Uustalu-Vene, *The Dual of Substitution is Redecoration*, Trends in Functional Programming 2002. The comonadic account of dropout is in Gal-Ghahramani, *Dropout as a Bayesian Approximation*, ICML 2016.

Chapter 14. Fibred Categories, Dependent Types, and Variable-Type Learning. Grothendieck fibrations are developed in SGA1 (Grothendieck, 1971) and in Jacobs, *Categorical Logic and Type Theory* (Elsevier, 1999). Dependent optics are discussed in Clarke et al. (Compositionality 2024, op. cit.) and in Vertech, *Dependent Optics* (2023, arXiv:2204.09547). Self-supervised learning as a fibration problem is an original perspective developed in this monograph; related ideas appear in Balestrieri-LeCun, *Contrastive and Non-Contrastive Self-Supervised Learning Recover Global and Local Spectral Embedding Methods* (2022, arXiv:2205.11508).

Chapter 15. Double Categories, Traced Monoidal Structures, and Modular Architectures. Double categories are defined in Ehresmann (1963); see also Grandis-Paré, *Limits in double categories*, Cahiers de topologie et géométrie différentielle catégoriques 40 (1999). The traced symmetric monoidal category and the Int construction are in Joyal-Street-Verity, *Traced Monoidal Categories*, Mathematical Proceedings of the Cambridge Philosophical Society 119 (1996). Hypergraph categories are in Fong’s doctoral thesis (Oxford, 2016) and Fong-Spivak, *Hypergraph Categories*, Journal of Pure and Applied Algebra 223 (2019). The double-categorical account of Para and backpropagation is Gavranović’s doctoral thesis (2024, op. cit.).

Chapter 16. Higher Categories, Coherence, and the Future of the Theory. The theory of bicategories is in Bénabou (1967). Tricategories are Gordon-Power-Street (1995). Quasi-categories ($(\infty, 1)$ -categories) are developed in Joyal’s extensive unpublished work and in Lurie’s *Higher Topos Theory* (Princeton, 2009; freely available) and *Higher Algebra* (2017, freely available). Homotopy type theory and the Univalence Axiom are in the *Homotopy Type Theory* book (The Univalent Foundations Program, IAS, 2013; freely available at homotopytypetheory.org). Formal verification of deep learning in Coq is pursued in several research groups; see e.g. Bagnall-Stewart (2019).

Chapter 17. Operads, Multicategories, and the Algebra of Network Wiring. Operads are introduced in May, *The Geometry of Iterated Loop Spaces* (Springer, 1972).

Coloured operads (symmetric multicategories) are in Lambek (1969). The little n -disks operads and their role in iterated loop spaces are covered in Boardman-Vogt (1973) and Adams (1978). The Boardman-Vogt tensor product and the theorem $\mathcal{D}_1 \otimes_{\text{BV}} \mathcal{D}_1 \simeq \mathcal{D}_2$ are in Boardman-Vogt (1973). Spivak's wiring diagrams are in *The operad of wiring diagrams*, Journal of Pure and Applied Algebra 219 (2015). The operadic perspective on neural networks is pursued in Shiebler-Gavranović-Wilson, *Category Theory in Machine Learning*, 2021 (arXiv:2106.07032).

Chapter 18. Equivariant Architectures and Categorical Symmetry. The geometric deep learning blueprint is Bronstein-Bruno-Cohen-Veličković, *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges* (2021, arXiv:2104.13478). The G-CNN architecture is Cohen-Welling, *Group Equivariant Convolutional Networks*, ICML 2016. Steerable CNNs are Cohen-Welling, ICPR 2017, and Weiler-Cesa, *General $E(2)$ -Equivariant Steerable CNNs*, NeurIPS 2019 (arXiv:1911.08251). The Peter-Weyl theorem is classical; see Bröcker-tom Dieck, *Representations of Compact Lie Groups* (Springer, 1985). The categorical treatment of G -sets as functor categories is in Mac Lane (1998), Chapter I.

Chapter 19. Information Geometry and the Natural Gradient Categorically. Amari's information geometry is in *Information Geometry and Its Applications* (Springer, 2016) and the classic *Methods of Information Geometry* with Nagaoka (AMS, 2000). The natural gradient is Amari, *Natural Gradient Works Efficiently in Learning*, Neural Computation 10 (1998). The categorical formulation of dual connections is in Ay-Jost-Lê-Schwachhöfer, *Information Geometry* (Springer, 2017). The Cramér-Rao bound in categorical form is related to the data processing inequality for f -divergences; see Raginsky, *Strong data processing inequalities and Φ -Sobolev inequalities for discrete channels*, IEEE Transactions on Information Theory 62 (2016).

Chapter 20. Categorical Signal Processing. The characterisation of translation-equivariant operators as convolutions is classical Fourier analysis; see Stein-Weiss, *Introduction to Fourier Analysis on Euclidean Spaces* (Princeton, 1971). The Fourier transform as a monoidal functor is discussed in Anel-Joyal, *Sweedler Theory for (Co)algebras and the Bar-Cobar Constructions* (2013). Spectral graph theory is in Chung, *Spectral Graph Theory* (AMS, 1997). GCNs are Kipf-Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, ICLR 2017 (arXiv:1609.02907). The scattering transform is Mallat, *Group Invariant Scattering*, Communications on Pure and Applied Mathematics 65 (2012), 1331–1398. Message passing as a natural transformation is formalised in Gavranović et al. (ESOP 2022, op. cit.).

Chapter 21. The Free Energy Principle, Active Inference, and Categorical Cybernetics. The FEP is due to Friston; the primary reference is Friston et al., *A free energy principle for the brain*, Journal of Physiology-Paris 100 (2006), 70–87. The categorical formulation of active inference is Capucci-Gavranović-Hedges-Rischel, *Towards Foundations*

of *Categorical Cybernetics* (MFPS 2022, op. cit.) and Sajid-Parr-Friston, *Active Inference: A Process Theory*, Neural Computation 2017. Categorical thermodynamics connecting free energy and Markov categories is in Baez-Fong, *A Compositional Framework for Markov Processes*, Journal of Mathematical Physics 57 (2016). Predictive coding is Rao-Ballard, *Predictive coding in the visual cortex*, Nature Neuroscience 2 (1999).

Chapter 22. Loss Functions, Divergences, and the Geometry of Learning Objectives. Proper scoring rules are Savage (1971) and Gneiting-Raftery, *Strictly Proper Scoring Rules, Prediction, and Estimation*, Journal of the American Statistical Association 102 (2007). The correspondence between proper losses and Bregman divergences is Banerjee-Guo-Wang, *On the Optimality of Conditional Expectation as a Bregman Predictor*, IEEE Transactions on Information Theory 51 (2005). f -divergences are Ali-Silvey (1966) and Csiszár (1967). The PAC-Bayes framework is Shawe-Taylor-Williamson (1997) and McAllester (1999, 2003); for a recent survey see Guedj, *A Primer on PAC-Bayesian Learning*, 2019 (arXiv:1901.05353).

Chapter 23. Generative Models: GANs, Flows, and Diffusion. GANs: Goodfellow et al., NeurIPS 2014 (arXiv:1406.2661). WGAN: Arjovsky-Chintala-Bottou, *Wasserstein Generative Adversarial Networks*, ICML 2017 (arXiv:1701.07875). RealNVP: Dinh-Sohl-Dickstein-Bengio, ICLR 2017 (arXiv:1605.08803). Glow: Kingma-Dhariwal, NeurIPS 2018 (arXiv:1807.03039). DDPM: Ho-Jain-Abbeel, NeurIPS 2020 (arXiv:2006.11239). Score-based models and the SDE framework: Song-Sohl-Dickstein-Kingma-Kumar-Ermon-Poole, ICLR 2021 (arXiv:2011.13456). Anderson’s reverse-time SDE: Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982). Score matching: Hyvärinen, *Estimation of Non-Normalized Statistical Models by Score Matching*, JMLR 6 (2005). Flow Matching: Lipman-Chen-Ben-Hamu-Nickel-Le, ICLR 2023 (arXiv:2210.02747); Albergo-Vanden-Eijnden, *Building Normalizing Flows with Stochastic Interpolants*, ICLR 2023.

Chapter 24. Residual Architectures, Skip Connections, and Categorical Constructions. ResNets: He-Zhang-Ren-Sun, *Deep Residual Learning for Image Recognition*, CVPR 2016 (arXiv:1512.03385). Pre-activation ResNets: He et al., ECCV 2016 (arXiv:1603.05027). DenseNets: Huang-Liu-van der Maaten-Weinberger, CVPR 2017 (arXiv:1608.06993). Highway Networks: Srivastava-Greff-Schmidhuber, NeurIPS 2015. Neural ODEs: Chen-Rubanova-Bettencourt-Duvenaud, NeurIPS 2018 (arXiv:1806.07366); the continuous-depth limit perspective is E, *A Proposal on Machine Learning via Dynamical Systems*, Communications in Mathematics and Statistics 5 (2017). The adjoint method for ODEs is Pontryagin et al., 1962; for Neural ODEs see Chen et al. (2018). Layer Normalisation: Ba-Kiros-Hinton (arXiv:1607.06450, 2016). SwiGLU: Shazeer (arXiv:2002.05202, 2020).

Chapter 25. Categorical Causal Inference and the Do-Calculus. Pearl’s do-calculus: Pearl, *Causality: Models, Reasoning, and Inference* (Cambridge, 2000, 2nd

ed. 2009), and the original papers Pearl (1995), Pearl-Robins (1995). SCMs and potential outcomes: Peters-Janzing-Schölkopf, *Elements of Causal Inference* (MIT Press, 2017; freely available). Completeness of the do-calculus: Shpitser-Pearl, *Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models*, AAAI 2006; Huang-Valtorta, *Pearl's Calculus of Intervention Is Complete*, UAI 2006. Twin networks and counterfactuals: Balke-Pearl, *Counterfactual Probabilities*, UAI 1994. Categorical causal inference: Fong, *Causal Theories: A Categorical Perspective on Bayesian Networks*, Master's thesis (Oxford, 2013; arXiv:1301.6201); Jacobs-Kissinger-Zanasi, *Causal Inference by String Diagram Surgery*, FoSSaCS 2019 (arXiv:1811.08338); Bongers-Forré-Peters-Mooij, *Foundations of Structural Causal Models with Cycles and Latent Variables*, Annals of Statistics 49 (2021). Causal representation learning: Schölkopf et al., *Toward Causal Representation Learning*, Proceedings of the IEEE 109 (2021); IRM: Arjovsky et al., arXiv:1907.02893.

Chapter 26. Meta-Learning, MAML, and Higher-Order Categorical Structures. MAML: Finn-Abbeel-Levine, *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*, ICML 2017 (arXiv:1703.03400). FOMAML and Reptile: Nichol-Achiam-Schulman, *On First-Order Meta-Learning Algorithms* (arXiv:1803.02999, 2018). Meta-SGD: Li et al., arXiv:1707.09835. Prototypical Networks: Snell-Swersky-Zemel, NeurIPS 2017 (arXiv:1703.05175). Matching Networks: Vinyals et al., NeurIPS 2016. Hypernetworks: Ha-Dai-Le, ICLR 2017 (arXiv:1609.09106). The 2-categorical framework for meta-learning as described here is original; related ideas are in Zhu et al., *Learning to Learn with Feedback and Local Plasticity*, Nature Machine Intelligence 2020.

Chapter 27. The Categorical Theory of Optimisation. Gradient descent convergence for convex functions: Nesterov, *Lectures on Convex Optimization* (Springer, 2018). Langevin dynamics and MCMC: Welling-Teh, *Bayesian Learning via Stochastic Gradient Langevin Dynamics*, ICML 2011. Wasserstein gradient flows: Ambrosio-Gigli-Savaré, *Gradient Flows in Metric Spaces and in the Space of Probability Measures* (Birkhäuser, 2005). Heavy ball method: Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics 4 (1964). Nesterov's accelerated gradient: Nesterov (1983). Underdamped Langevin and hypocoercivity: Villani, *Hypocoercivity*, Memoirs of the AMS 202 (2009). Adam: Kingma-Ba, ICLR 2015 (arXiv:1412.6980). K-FAC: Martens-Grosse, ICML 2015 (arXiv:1503.05671). Shampoo: Gupta-Koren-Singer, ICML 2018. Flat minima and generalisation: Hochreiter-Schmidhuber, *Flat Minima*, Neural Computation 9 (1997); Keskar et al., *On Large-Batch Training for Deep Learning*, ICLR 2017 (arXiv:1609.04836). SAM: Foret et al. (2021, op. cit.).

Chapter 28. Graph Neural Networks and the Categorical Geometry of Relational Data. The category of graphs and its presheaf structure: see Mac Lane-Moerdijk (1992), Chapter I. MPNNs: Gilmer-Schütt-Riley-Wiedemann-Müller-Lilienfeld, *Neural Message Passing for Quantum Chemistry*, ICML 2017 (arXiv:1704.01212). The WL test and GNN expressivity: Weisfeiler-Lehman (1968); Xu-Hu-Leskovec-Jegelka, *How Powerful*

are *Graph Neural Networks?*, ICLR 2019 (arXiv:1810.00826); Maron-Ben-Hamu-Shamir-Lipman, *Invariant and Equivariant Graph Networks*, ICLR 2019 (arXiv:1812.09902). Spectral GCNs: Bruna et al. (ICLR 2014); Defferrard-Bresson-Vandergheynst, NeurIPS 2016; Kipf-Welling (ICLR 2017, arXiv:1609.02907). Graph Transformers: Kreuzer et al., *Rethinking Graph Transformers with Spectral Attention*, NeurIPS 2021; Rampásek et al., *Recipe for a General, Powerful, Scalable Graph Transformer*, NeurIPS 2022. DiffPool: Ying et al., NeurIPS 2018. MinCutPool: Bianchi et al., ICML 2020.

Chapter 29. Transformers and the Category of Sequence-to-Sequence Maps.

The original Transformer: Vaswani-Shazeer-Parmar-Uszkoreit-Jones-Gomez-Kaiser-Polosukhin, *Attention Is All You Need*, NeurIPS 2017 (arXiv:1706.03762). BERT: Devlin-Chang-Lee-Toutanova, NAACL 2019 (arXiv:1810.04805). GPT-2/3: Radford et al. (2019); Brown et al., NeurIPS 2020. The memory view of FFN layers: Geva-Schuster-Berant-Levy, *Transformer Feed-Forward Layers Are Key-Value Memories*, EMNLP 2021. Relative positional encoding: Shaw-Uszkoreit-Vaswani, NAACL 2018; Dai et al. (Transformer-XL), ACL 2019. RoPE: Su-Ahmed-Lu-Pan-Bo-Liu (arXiv:2104.09864). Linear attention: Katharopoulos et al., ICML 2020; Performer: Choromanski et al., ICLR 2021 (arXiv:2009.14794). Turing completeness of Transformers: Pérez-Marinković-Barceló, *On the Turing Completeness of Modern Neural Network Architectures*, ICLR 2019 (arXiv:1901.03429). Profunctor composition view of attention: implicit in Gavranović (2024, op. cit.).

Chapter 30. Reinforcement Learning, Markov Decision Processes, and Categorical Dynamics.

MDPs and dynamic programming: Bellman, *Dynamic Programming* (Princeton, 1957). DQN: Mnih et al., *Human-level control through deep reinforcement learning*, Nature 518 (2015). Policy gradient theorem: Sutton-McAllester-Singh-Mansour, NeurIPS 1999. Natural policy gradient: Kakade, NeurIPS 2002. Actor-critic: Kondatsitsiklis, NeurIPS 2000. PPO: Schulman et al. (arXiv:1707.06347, 2017). RLHF: Christiano et al., NeurIPS 2017; Ziegler et al. (arXiv:1909.08593, 2019); Stiennon et al., NeurIPS 2020. DPO: Rafailov et al., NeurIPS 2023 (arXiv:2305.18290). The categorical account of MDPs as polynomial coalgebras is original; related ideas appear in Hedges, *Morphisms of Open Games*, Electronic Notes in Theoretical Computer Science 341 (2018).

Chapter 31. Neural Scaling Laws and the Category of Model Families. The original scaling laws: Kaplan-McCandlish-Henighan-Brown-Chess-Child-Gray-Radford-Wu-Amodei, *Scaling Laws for Neural Language Models*, arXiv:2001.08361 (2020). Chinchilla laws: Hoffmann-Borgeaud-Mensch-Buchatskaya-Cai-Rutherford-Casas-Hendricks-Welbl-Clark-Hennigan-Noland-Millican-van den Driessche-Damoc-Guy-Osindero-Simonyan-Elsen-Rae-Vinyals-Sifre, *Training Compute-Optimal Large Language Models*, NeurIPS 2022 (arXiv:2203.15556). Emergent capabilities: Wei et al., *Emergent Abilities of Large Language Models*, Transactions on Machine Learning Research 2022. Theoretical accounts of scaling: Bahri-Dyer-Kaplan-Lee-Sharma, *Explaining Neural Scaling Laws*, PNAS 2024 (arXiv:2102.06701); Sharma-Kaplan (arXiv:2203.06875). Rate-distortion and scaling: Bordelon-Canatar-Pehlevan, *Spectrum Dependent Learning Curves in Kernel Regression and Wide Neural Networks*, ICML 2020.

Chapter 32. Categorical Linear Algebra, Tensor Networks, and Quantum-Inspired Methods. Matrix product states and tensor trains: Oseledets, *Tensor-Train Decomposition*, SIAM Journal on Scientific Computing 33 (2011). HOSVD and Tucker: De Lathauwer-De Moor-Vandewalle, SIAM Journal on Matrix Analysis and Applications 21 (2000). MERA: Vidal, *Entanglement Renormalization*, Physical Review Letters 99 (2007). Tensor decomposition in CNNs (Tucker for convolutions): Kim et al., ICLR 2016. MobileNet: Howard et al. (arXiv:1704.04861). Quantum-inspired algorithms: Tang, *A quantum-inspired classical algorithm for recommendation systems*, STOC 2019; Chia et al., *Sampling-based sublinear low-rank matrix arithmetic framework*, STOC 2020. The compact closed structure of $\mathbf{FVect}_{\mathbb{R}}$ and its relation to quantum computing: Abramsky-Coecke, *A categorical semantics of quantum protocols*, LICS 2004. Partial traces in machine learning: related to the Gram matrix computation in kernel methods; see Schölkopf-Smola (2002, op. cit.).

Chapter 33. Categorical Semantics of Programming Languages and Differentiable Programming. The Curry-Howard correspondence: Curry (1934), Howard (1969). The Lambek-Scott theorem: Lambek-Scott (1986, op. cit.). Linear logic: Girard, *Linear Logic*, Theoretical Computer Science 50 (1987). Linear types and resource management: Wadler, *Linear Types Can Change the World*, IFIP TC 2 (1990). AD as a functor between type theories: Huot-Staton-Vákár, *Correctness of Automatic Differentiation via Diffeologies and Categorical Gluing*, FoSSaCS 2020 (arXiv:2001.02209); Cruttwell-Gavranović-Ghani-Wilson-Zanasi (ESOP 2022, op. cit.); Vákár, *Reverse AD at Higher Types*, FoSSaCS 2021. Zygote: Innes et al., *Don't Unroll Adjoint: Differentiating SSA-form Programs*, arXiv:1810.07951 (2019). JAX: Bradbury et al., *JAX: composable transformations of Python+NumPy programs*, 2018. Abstract interpretation for neural network verification: Gehr-Mirman-Drachsler-Cohen-Tsankov-Chaudhuri-Vechev, *AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation*, IEEE S&P 2018; Singh-Gehr-Püschel-Vechev, *An Abstract Domain for Certifying Neural Networks*, POPL 2019.